Application de deux méthodes de télédétection rapprochée à l'étude des escarpements rocheux instables : la photogrammétrie et la scannerisation laser

Bruno Fricout

LGCA

UNIVERSITE CHAMBERY ANNECY SAVOIE

Ecole doctorale Terre Univers Environnement

THESE PRESENTEE POUR OBTENIR LE TITRE DE :

Docteur *ès* Sciences, spécialité Géologie

Application de deux méthodes de télédétection rapprochée à l'étude des escarpements rocheux instables : la photogrammétrie et la scannerisation laser

PAR :

Bruno Fricout

Membres du Jury :

Jacques Angelier - Professeur, Université Pierre et Marie Curie, Paris VI . Rapporteur,
Benoit Deffontaines – Professeur, Université de Paris Est, Marne-la-Vallée Rapporteur
Denis Jongmans – Professeur, Université Joseph Fourier, Grenoble. Examinateur,
Fluvio Rinaudo – Professeur, Politecnico de Turin. Examinateur,
Frédéric Donzé – Professeur, Université Joseph Fourier, Grenoble. Examinateur,
Thierry Villemin – Professeur, Université de Savoie. Directeur de thèse,

À l'heure actuelle, l'étude d'une falaise instable nécessite la mesure sur place, à l'aide d'une boussole clinomètre, de sa fracturation. Ce n'est pas sans inconvénient pour des raisons d'accessibilité et de dangerosité de la mesure.

Pour étudier la morphologie de ces falaises sans recourir à ces mesures in-situ, il faut faire appel à des méthodes de télédétection. Deux méthodes semblent à priori particulièrement adaptées pour ce faire. La photogrammétrie permet de calculer la position XYZ d'un point à partir de la position de l'image de ce point sur deux photographies prisent depuis des points de vue différents. Cette technique éprouvée connaît un nouvel essor avec les progrès des appareils numériques. La scannerisation laser, quant à elle, consiste à calculer la position d'un très grand nombre de points de la paroi en mesurant le temps de trajet aller-retour d'une impulsion laser. Ces deux méthodes connaissent des limitations différentes en terme de résolution, de précision, de temps d'acquisition sur site et de traitement à postériori.

La photogrammétrie et la scannerisation laser ont été utilisées sur quatre sites afin de tester l'efficacité de ces méthodes sur des cas concrets de falaise instable. Le site du Rocher du Midi (Vercors, 38, France) a été étudié à l'aide de deux scans laser réalisés depuis le sommet de la paroi instable, avec des points de vue très rasants sur la zone d'étude, ainsi qu'au moyen d'une couverture photographique réalisée avec un appareil argentique grand format depuis un hélicoptère. Le Ravin de L'aiguille (Chartreuse, 38, France) a été étudié à partir de deux scannerisations laser réalisées en tête de paroi, l'une depuis un point de vue très rasant et l'autre depuis un point de vue presque frontal, permis par la géométrie très échancrée de la ligne de crête, ainsi qu'avec une couverture photographique réalisée avec un réflex numérique depuis un hélicoptère. Le Rocher de la Bourgeoise (Vercors, 38 France) a été étudié au moyen d'une scannerisation laser héliportée et d'une couverture photographique réalisée en même temps avec un appareil numérique. Les gorges de Paganin (06, France) ont été étudiées au moyen de trois scannerisations laser réalisées depuis le pied de la paroi.

Pour pouvoir exploiter les données de scannerisation laser issues de ces quatre sites, le concept d'image solide, développé par le Politecnico de Turin, a été réimplanté dans le logiciel ImageJ afin de développer aisément des greffons pour chacune des mesures utiles à l'exploitation. La comparaison des résultats obtenus par cette méthode avec ceux obtenus par photogrammétrie et ceux issus des mesures directe sur site valide l'efficacité des deux méthodes de télédétection, avec cependant quelques réserves. les méthodes de télédétection, et en particulier l'image solide, apportent une plus value notable par rapport aux mesures in-situ en terme de capacité de support aux mesures géophysiques complémentaires et de réalisation d'un modèle géométrique complet en prélude à une simulation de stabilité.

Cependant des développement restent à faire, tant sur l'exploitation des scannerisations laser que sur l'utilisation de la photogrammétrie, pour permettre de tirer le meilleur parti possible de ces deux techniques.

Today, studying an unstable cliff requires on-site fracture measurement by means of a clinometer compass. From accessibility and safety point of view, this method is far from perfect.

Teledetection techniques provides an alternative for studying the cliffs morphology without such in-place measurements. Two methods seams beforehand particularly well suited. Photogrammetry on the one hand leads to the computation of the XYZ position of a point starting from its 2D position in two photographs taken from slightly different point of view. Numeric cameras have brought great improvements to this robust technique. Laser scanning on the other hand, consist in computing the position of a large number of points on the wall by measuring the time used by a laser impulse to reflect on the cliff. These two methods are subject two different limitations in terms of resolution, precision, on-site acquisition time and required computing power.

Photogrammetry and laser scanning have been used on four sites to test their efficiency on real unstable cliffs. The Rocher du Midi zone (Vercors, 38, France), has been first studied with two laser scans acquired with a very grazing point of view from the top of the unstable wall. It has also been photographed with a large field silver camera from an helicopter. The Ravin de l'aiguille (Chartreuse, 38, France) has been studied through two laser scans realized ahead from the wall. The first one at grazing angle and the other one almost from a front point of view. These acquisitions were possible thanks to the crest line specific geometry. A photographic cover has also been realized from a numeric reflex on helicopter board. The Rocher de la bourgeoise (Vercors, 38, France), has been measured by means of a laser scanning on helicopter board and by mean of pictures taken at the same time from a numeric camera. The Gorges de Paganin (06, France) have been studied with three laser scan from the bottom of the cliff.

To be able to use laser scan data from these four sites, the concept of solid image developed by the Turin Polotecnico has been reimplemented in the ImageJ software. Thus, add-ons have been easily developed for each measurement usefull for the exploitation. Comparing the results obtained by this technique with those obtained with photogrammetry or direct on-site measurement have proved, with several restrictions, the efficiency of both teledection methods. Teledetection in general and the solid image concept in particular provide a noticeable added value compared to on-site measurements because they can be used as a support to complementary geophysic measurements or they can help the computation of a full geometric model preliminary to stability simulation.

Despite these good results, improvements could still be obtained to exploit at best the laser scan and the photogrammetry techniques.

Table des matières

Table des figures

Liste des tableaux

Un besoin de méthodes d'études à distantes des parois rocheuses

Lorsque l'étude d'un escarpement rocheux supposé instable est réalisée, la mesure des plans de fracturation est généralement réalisée sur place à l'aide d'une boussole clinomètre. Cette méthode,très largement utilisée jusqu'à présent, possède des inconvénients. Certains sont systématiques et d'autres se font ou non sentir, en fonction des particularités du site étudié.

Manque de précision et de complétude des mesures

Lors d'un relevé de fracture réalisé à plat, la localisation dans l'espace des discontinuités échantillonnées peut être précisée imparfaitement à l'aide d'un décamètre. Ceci suffit en général pour estimer les paramètres de densité et d'espacement. Lors d'un relevé de fracture en paroi, effectué dans des conditions acrobatiques, la position des fractures les unes par rapport aux autres ne peut être estimée, quand elle l'est, que de manière très approximative.

En plus de la difficulté à localiser les discontinuités, qui se traduit souvent par une absence totale de mesure de position, la mesure de l'orientation du plan lors d'une

descente en rappel est difficile à réaliser pratiquement, surtout si la paroi présente des surplombs ou des dévers. Dans ces conditions, le plongement des plans subverticaux est rarement mesurable et leur direction n'est pas mesurée avec autant de facilité que lors d'un relevé effectué depuis le sol.

Enfin les descentes sur corde ne peuvent couvrir toute la surface de la paroi. Seuls les discontinuités situées le long de la corde pourront être mesurés. Cette absence de couverture latérale peut induire un biais statistique sur les fréquences des fractures si la répartition des fractures n'est pas homogène. Elle représente aussi le risque de ne pas remarquer une discontinuité qui, par sa position et ses caractéristiques (ouverture, continuité, etc.), joue un rôle majeur dans la stabilité du site.

Parois difficilement accessibles

En plus de ces problèmes liés à la nature du relevé de fracture en paroi et qui sont systématiquement présents, on rencontre parfois des problèmes liés aux difficultés d'accès. Sur certains sites, ce problème pourra être absent, par exemple lorsqu'une route passe à proximité du sommet. Sur d'autres il aura une influence sur le coût des études sans avoir de réelle influence sur leur faisabilité. Par exemple, Sur le site du Rocher du Midi que nous étudierons par la suite, une heure de portage était nécessaire entre la route et le sommet de la falaise. Certains escarpements en haute montagne pose même un problème de faisabilité de l'étude, lorsque l'accès nécessite une dépose hélicoptère par exemple.

Parois dangereuses

Le problème de la sécurité se pose régulièrement pour les personnes devant réaliser les relevés en paroi. Les sites où des éboulements sont à craindre sont aussi, souvent, des sites sur lesquels les départs de blocs sont fréquents. Le port du casque n'est pas toujours une garantie de sécurité suffisante pour les personnes devant réaliser les études de paroi. Dans ce cas le seul moyen pour disposer d'informations structurales sur la paroi étudiée sont soit l'extrapolation de mesures réalisées à proximité, soit les méthodes d'études à distance.

Impératifs de fonctionnement d'équipements proches

Enfin les mesures directes sur la paroi peuvent être rendues dangereuses par la présence d'infrastructures en pied de paroi pour lesquels une interruption de fonctionnement n'est pas prévu à ce stade de l'étude. C'est fréquemment le cas des routes ou des voies ferrées passant à proximité de paroi, et dont on veut garder l'usage tant que l'aléa n'a pas été clairement diagnostiqué.

Pour toutes ces raisons, le besoin d'étudier les escarpement rocheux par des méthodes de télédétection se fait clairement sentir. Afin de déterminer dans quelles conditions elles peuvent se substituer aux mesures traditionnelles, nous avons testé l'applicabilité de deux méthodes qui ont bénéficier de récents progrès technologiques ou instrumentaux : la scanneristaion laser et la photogrammétrie numérique.

Chapitre 1

Photogrammétrie et scannerisation laser, deux méthodes actuelles de télédétection rapprochée

1.1 Définitions

La *télédétection* désigne la mesure ou l'acquisition d'informations sur un objet d'étude faite à distance, sans qu'il y ai contact entre l'instrument de mesure et l'objet étudié. L'interaction entre outil et objet de mesure peut être basé sur les ondes sonores, les ondes sismiques, les ondes électromagnétiques, les ondes lumineuses et même sur la gravité. Les formes d'interaction à distance sont très variées, chacune ayant ses instruments et sa discipline propre, même si certaines méthodes peuvent être généralisées à plusieurs disciplines.

La télédétection est très utilisée dans le domaine militaire. De grandes avancées dans

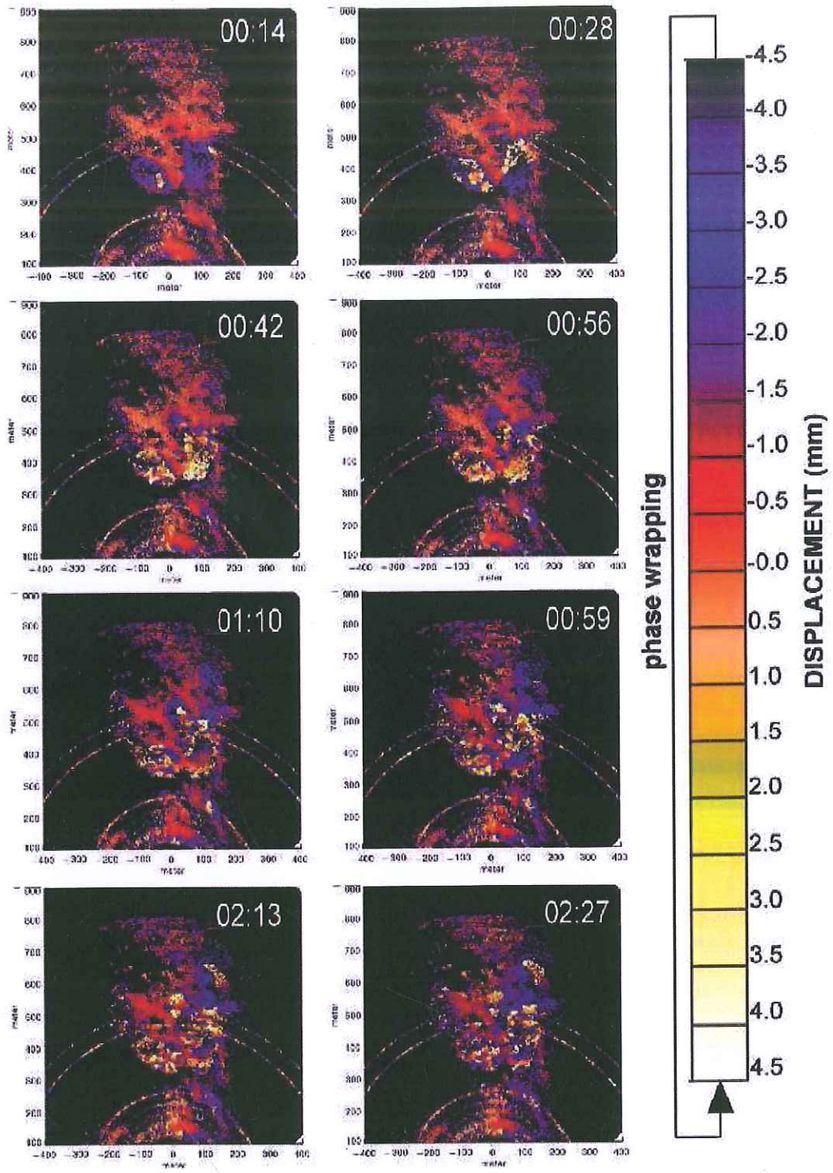


Fig. 1.1 – Exemple de suivi de versant par interférométrie radar.
Séquence d'interferogrammes réalisée sur le glissement de Tessine (Italie) entre 00h14 et 02h27 le 05-10-2000. D'après Tarchi et al. *(2003).*

les domaines du sonar et du radar, par exemple, sont issues de travaux de recherche militaires. La télédétection est aussi utilisée, depuis des satellites, afin d'étudier l'atmosphère ou la surface terrestre avec une large couverture. On parle alors de *télédétection satellitaire*. Les satellites ERS (radar) ou Spot(imagerie visible et IR haute résolution) en sont de bonnes illustrations. La télédétection peut aussi être utilisée sur des objets proches, avec alors une plus grande résolution. On parle alors de *télédétection rapprochée*. C'est le cas par exemple de l'interferométrie radar avec un radar fixe pour détecter des mouvement de faible amplitude sur des versants instables (fig. 1.1).

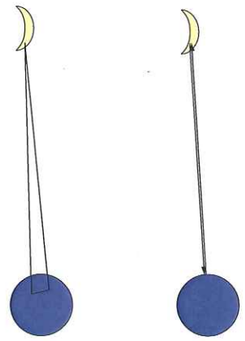


Fig. 1.2 – Différences entre télédétection passive et active. Exemple de la mesure de la distance Terre-Lune dans le domaine de la lumière visible.
À gauche de manière passive : la position est déterminée à partir de la mesure de l'angle de visée. Deux points de vue différents sont nécessaires pour déterminer la position, mais c'est le soleil, source d'énergie naturelle, qui éclaire la lune sans dépense d'énergie supplémentaire. À droite de manière active : un rayon laser est émis vers la lune et réfléchi par un prisme installé sur place. Un seul instrument permet de déterminer la distance en calculant le temps de trajet aller-retour du rayon laser. Le prisme installé sur la lune permet de réduire l'énergie nécessaire à la mesure en renvoyant toute l'énergie dans la même direction.

Dans la télédétection, on distingue les méthodes passives et les méthodes actives (fig. 1.2). Pour la *télédétection active*, une onde est envoyée vers l'objet d'étude depuis une source artificielle et c'est le retour de cette onde qui donne des informations sur l'objet ; pour la *télédétection passive*, c'est une source naturelle, souvent l'objet d'étude lui même, qui émet des ondes qui sont perçues par l'instrument de mesure. Les méthodes

passives sont nettement moins consommatrices d'énergie que les méthodes actives correspondantes. Les méthodes actives sont par contre plus efficaces pour déterminer la position de l'objet étudié. En effet les méthodes actives permettent de mesurer le temps de trajet aller retour de l'onde, ce que ne permettent pas les méthodes passives.

Lorsque la vitesse et la direction de l'onde peuvent être considérées comme constantes, une méthode passive peut déterminer la position de l'objet ou de la partie de l'objet étudié en utilisant deux récepteurs éloignés les uns des autres. Dans le domaine de l'imagerie on parle de *principe de stéréoscopie*. Ce principe est utilisé en photogrammétrie et sera détaillée au chapitre 1.2. Dans les mêmes conditions, la méthode active correspondante, le lidar (cf. chapitre 1.3.1), permet de déterminer la position dans l'espace avec un seul instrument (fig. 1.2).

Quand la propagation n'est pas rectiligne, ou n'est pas à vitesse constante, il faut disposer d'un modèle de vitesse afin de déterminer la trajectoire des ondes (fig. 1.3). Ce modèle de vitesse peut être acquis préalablement et séparément. C'est le cas des sondages bathythermiques, qui permettent d'établir un modèle de propagation des ondes sonores dans l'eau. Dans ce cas un seul instrument de télédétection active est nécessaire pour établir la distance et la position de l'objet détecté, un seul sonar dans l'exemple indiqué. Le modèle de propagation peut aussi parfois être calculé grâce aux mesures elles même, C'est le principe de *la tomographie*. Il faut alors de nombreux émetteurs et récepteurs synchronisés pour réaliser une inversion de données (fig.1.4). Même si les modèles mathématiques utilisés pour réaliser l'inversion des données de tomographie ont une influence notable sur la qualité des résultats [Natterer, 2001], l'inversion est toujours d'autant plus précise que le réseau d'émetteurs récepteurs est dense.

L'amélioration du diagnostic de stabilité des falaises rocheuses, qui est l'objet final

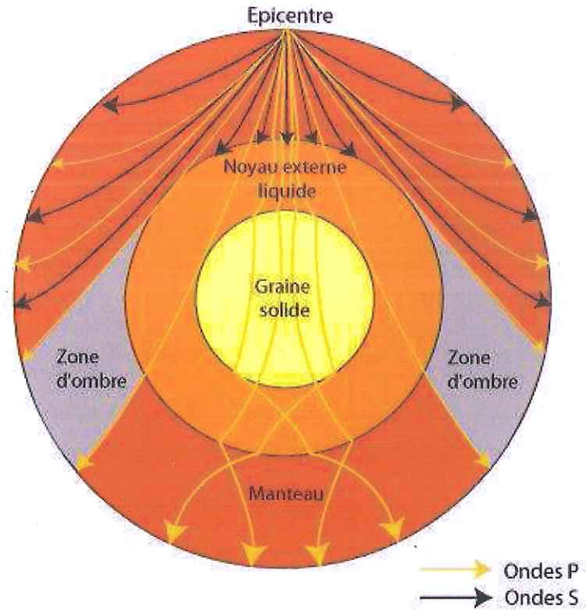


FIG. 1.3 – Trajet des ondes sismiques au travers de la terre après un séisme.
La vitesse des ondes sismiques dépend des paramètres physiques des roches traversées(densité, module d'incompressibilité, module de cisaillement). La variation de ces paramètres en fonction de la profondeur rend les trajectoires courbes. Un modèle de vitesse des ondes en fonction de la profondeur est nécessaire pour localiser la source.

de ce travail, implique de résoudre deux problèmes pour lesquels la télédétection est, sinon irremplaçable, du moins d'un intérêt majeur :

- La détermination de la géométrie extérieure. La mesure in situ de cette géométrie pose des problèmes pour les raisons exposées dans le chapitre introductif.
- La détermination des caractéristiques à l'intérieur du massif. Une détermination par sondage est généralement impossible à cause de l'instabilité du massif et des difficultés à mener des travaux en paroi.

Le travail présenté dans ce manuscrit se limite au problème de la géométrie extérieure, celui des caractéristiques internes étant étudié par ailleurs, entre autre par l'équipe du Laboratoire de Géophysique Interne et Tectonophysique (D. Jongmans) avec qui nous

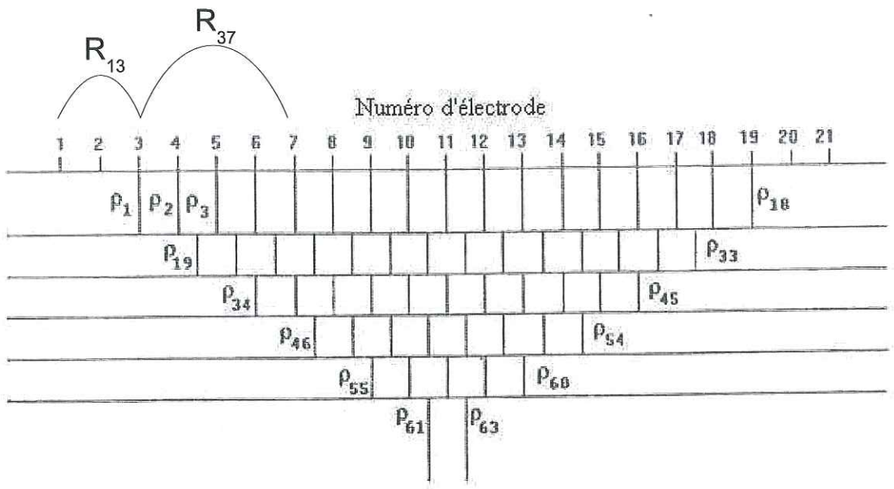


FIG. 1.4 – Principe de la tomographie électrique.
On mesure la résistance R_{ij} entre les différentes électrodes. Celle-ci dépend de la résistivité ρ_{ij} entre deux électrodes, en surface mais aussi en profondeur. Le problème direct, passer des résistivités locales aux résistances mesurées, peut être résolu analytiquement. Le passage des résistances mesurées aux résistivités locales est un problème d'inversion de données qui ne peut être résolu que de proche en proche par itérations successives.

avons collaboré sur l'étude de trois sites (Rocher du Midi, Ravin de l'Aiguille et Rocher de la Bourgeoise (cf. Chapitre 2). Les bases des deux méthodes de télédétection qui ont été utilisées dans ce but sont présentées dans la suite du chapitre. Ils s'agit de deux méthodes basées sur la lumière visible. Une méthode passive, la photogrammétrie ; une méthode active, la scannerisation laser.

1.2 Photogrammétrie

1.2.1 Principes généraux

Egels (2002a) définit la photogrammétrie comme « l'ensemble des techniques de mesures permettant la modélisation d'un espace 3D en utilisant des images 2D ». La photogrammétrie est une technique presque aussi vielle que la photographie qui a connu un renouveau avec l'apparition, puis la démocratisation, de l'imagerie numérique. Cette technique présente des formes très variées. Dans sa forme la plus classique les coordonnées en trois dimensions des points d'un objet sont déterminées par des mesures faites sur plusieurs images prises à partir de positions différentes. Généralement l'image est une photographie au sens habituel du terme qu'on appelle aussi capteur plan, ou matriciel. Cependant dans certaines applications l'image est recomposée à partir d'un capteur linéaire.

Dans la plupart des applications on travaille sur un couple de photographie, c'est à dire sur deux photographies de la même zone prises depuis deux positions différentes. L'identification d'un même détail sur les deux photographies permet de calculer sa position dans l'espace (cf. 1.2.2). Ce calcul nécessite de connaître les positions et les orientations correspondant aux deux images (cf. 1.2.4) ainsi que les caractéristiques géométriques précises de l'appareil de prise de vue (cf. 1.2.6).

La photogrammétrie est utilisée dans une grande variété d'application. Ces situations peuvent être regroupées dans des catégories issues des trois système de classification suivant : selon la distance entre l'objet et le capteur, selon l'orientation de l'axe optique et selon la différence d'orientation entre les deux photographies. Le distance de prise de vue peut-être :

– De l'ordre de 10 000 km lorsque le porteur est un satellite. On est alors généra-

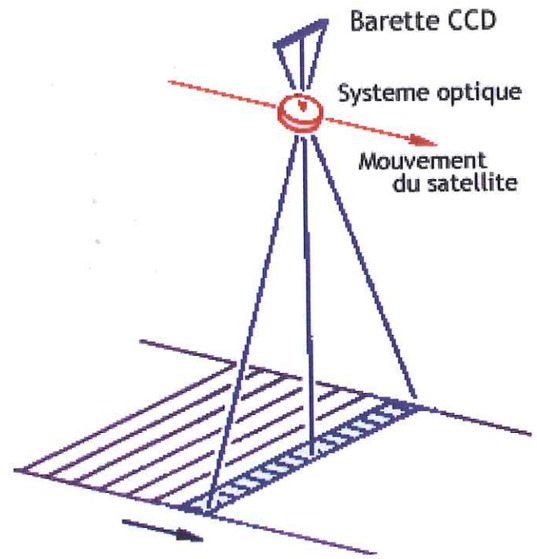


Fig. 1.5 – Principe de fonctionnement du capteurs linéaires.
Construction d'une image satellite à partir d'un capteur linéaire. À l'instant t le capteur enregistre la signature radiométrique de la ligne située en dessous de lui, à l'instant $t+\delta$ il enregistre une nouvelle ligne situé un peu plus en avant grâce aux déplacement du satellite.

lement dans une situation de photogrammétrie verticale, parallèle, comme dans le cas de l'avion.

- De l'ordre de plusieurs centaines à plusieurs milliers de mètres. C'est le cas des photographies aériennes. Dans ce cas on se trouvera également dans une situation de photogrammétrie verticale parallèle.
- De l'ordre de quelques dizaines à quelques centaines de mètres. C'est le cas de la photographie prise depuis un point fixe terrestre, réalisée, par exemple, à but architectural.
- Il est également possible d'utiliser la photogrammétrie pour étudier des phénomènes en laboratoire. Dans ce cas il faut ajuster la mise au point sur l'objet d'étude. L'appareil de prise de vue doit alors être calibré pour ce réglage de la distance focale.

L'orientation de l'axe optique peut-être :

- Verticale, vers le bas. C'est le cas le plus fréquent, car il correspond à l'utilisation de la photogrammétrie aérienne à des fins de cartographie. Certains logiciels commerciaux ne fonctionnent que dans ce cas de figure.
- Horizontale. C'est souvent le cas en photogrammétrie terrestre.
- Oblique, par exemple pour l'étude de monuments ou le faible recul implique des vues en contre plongée.

On différencie aussi les photogrammétries :

- Parallèles, lorsque les axes de prise de vue sont parallèles ou diffèrent de moins de 5° pour les deux photographies.
- Convergente lorsque ces deux axes sont différents.

La photogrammétrie parallèle permet d'utiliser la faculté de vision stéréoscopique : les deux photographies sont prises dans des conditions analogues à celles de nos yeux lorsque l'on regarde un objet proche. En utilisant un stéréoscope, système optique (par exemple un jeu de miroirs) faisant que chaque œil ne voit qu'une seule photographie à la

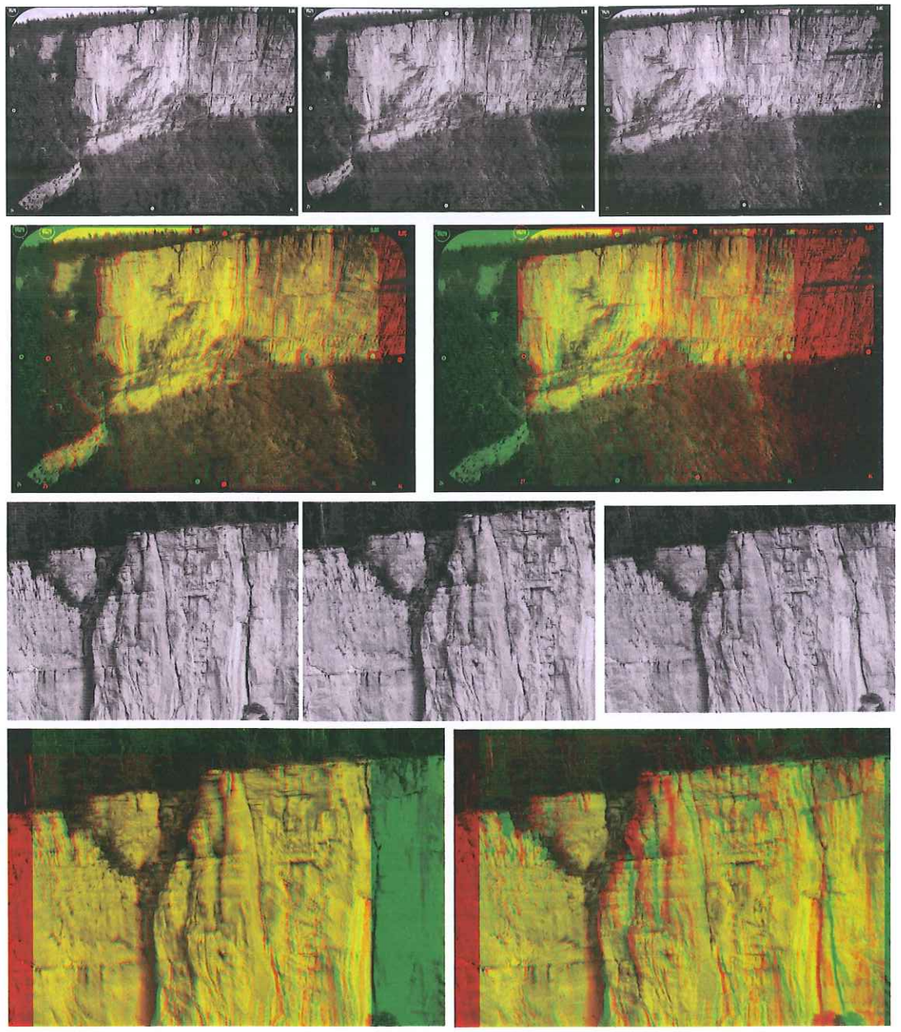


Fig. 1.6 – Exemples de couples stéréoscopique réalisées avec les photographies du rocher du midi.
En première ligne : De gauche à droite, image n° 24, 25 et 27. En deuxième ligne, couples composés des images n°24 et 25 (à gauche) et 24 et 27 (à droite). En troisième ligne, agrandissement d'une zone détaillée des photographies n° 24, 25 et 27. En quatrième ligne, agrandissement d'une zone détaillée des couples composés des images n°24 et 25 (à gauche) et 24 et 27 (à droite).
Lorsque la base photogramétrique (distance entre les deux points de vue) est petite (couple 24-25), la perception du relief est faible, mais le taux de recouvrement est important et la corrélation des détails est aisée. Lorsque la base photogramétrique est grande (couple 24-27), la perception du relief est forte, mais le taux de recouvrement est faible et la corrélation des détails est plusdifficile.

fois, un modèle de l'objet photographié est perçu en relief. Les conditions dans lesquelles cette observation est faite diffèrent des conditions de la vision naturelle. Le relief peut être ainsi perçu de manière exagérée. On qualifie ce phénomène d'*hyperstéréoscopie.*

Lorsque les deux points de vue sont proches l'un de l'autre, l'identification des points homologues est aisée car les détails se présentent sous le même aspect sur les deux photographies. Par contre la perception du relief est atténuée car l'hyperstéréoscopie est moins importante. À l'inverse, lorsque les points de vue sont éloignés, l'identification d'objets homologues est plus délicate, mais le relief est bien perçu et la précision sur la position mesurée est meilleure (fig. 1.6). De plus, le recouvrement entre les photographies (c'est à dire la portion d'image présente sur les deux photographies) diminue lorsque cette distance augmente, limitant les possibilités d'utiliser des points de vue très éloignés. La dimension de la base photogrammétrique généralement utilisée, c'est à dire l'écart entre les deux points de prise de vue, est généralement choisie équivalente à la distance séparant les point de vue de l'objet étudié.

La photogrammétrie convergente est surtout utilisée en architecture ou en urbanisme. C'est cette méthode qui permet de faire le tour d'un objet en utilisant un minimum de photographies. Comme les directions de prise de vue sont très différentes, il est impossible d'obtenir une vision stéréoscopique, en relief. Les objets ne présentent pas forcément les mêmes faces et n'ont donc pas le même aspect visuel d'une photographie à l'autre. Il faut donc que les points à mesurer soient aisément identifiables sur chaque photographie. C'est le cas des angles d'immeubles.

Un petit lexique des principaux termes de photogrammétrie, ainsi que des symboles utilisés dans ce mémoire est donné dans le tableau 1.1

Axe optique	Axe perpendiculaire au plan focal, passant par le milieu de l'axe optique. C'est la direction dans laquelle regarde l'appareil photo.
Corrélation	Similarité entre un détail de deux photographies. Deux points homologues ont une très forte corrélation, que ce soit visuellement, ou numériquement en calculant la ressemblance entre les voisinages respectifs des deux points.
Distorsion	Écart entre l'image obtenue sur le plan focal et l'image que l'on obtiendrait avec une lentille parfaite de même focale.
Épipolaire	Projection sur une image de l'ensemble P des points de l'espace correspondants à un point donné de l'autre image. P étant une droite de l'espace, l'épipolaire est une droite sur la photographie lorsque la distorsion est nulle.
Focale f	Distance de la lentille à laquelle converge tous les rayons provenant d'un objet situé à l'infini.
Marque de fond de chambre	Repère, souvent en forme de croix, situé au fond d'une chambre métrique et qui est impressionné sur le cliché au moment de la prise de vue. La position pixel des repères de fond de chambre permet de calculer l'orientation interne.
Orientation absolue	Détermination de la position et de l'orientation de l'appareil photographique au moment des deux prises de vue dans un repère terrain fixe.
Orientation interne	Détermination de la relation entre les coordonnées d'un point sur la photographie et ses coordonnées dans le plan focal de l'appareil photographique au moment de la prise de vue.
Orientation relative	Détermination, à une constante près, de la position et de l'orientation de l'appareil photographique au moment de la deuxième prise de vue dans le repère lié à la position de la première prise de vue.
Parallaxe	Décalage entre l'épipolaire d'un point calculée par les paramètres d'orientation et l'épipolaire réelle, lié à une légère erreur dans l'orientation relative ou dans le modèle de distorsion.
Plan focal	Plan ou se reprojette l'image d'un objet situé à l'infini de l'appareil photo. C'est le plan où se trouve le négatif argentique ou le capteur numérique pendant la prise de vue.
Point d'appui	Points dont on connait les coordonnées dans le repère terrain. Ces points servent à calculer l'orientation absolue du couple de photographies.
Points homologues	Les deux points, l'un sur la photographie de droite et l'autre sur la photographie de gauche, qui sont l'image d'un même point de l'espace.
Point principal ou point focal	Point du plan focal le plus proche de la lentille. C'est sur ce point que se trouve l'image d'un objet situé sur l'axe optique
Stéréoscopie, monoscopie	La stéréoscopie est la vision en relief procurée par le fait de regarder la photographie gauche avec l'œil gauche et la photographie droite avec l'œil droit. La monoscopie consiste à voir la même image avec les deux yeux.
$(X_{p1}\,;Y_{p1})\,;(X_{p2}\,;Y_{p2})$	Coordonnées pixel d'un point sur la photographie 1 et sur la photographie 2 (fig. 1.7).
$X_t\,;Y_t\,;Z_t$	Coordonnées dans le repère terrain, ou absolu, d'un point de l'espace (fig. 1.7).
$X_i\,;Y_i\,;Z_i$	Coordonnées d'un point de l'espace dans le repère lié à la position et l'orientation de l'appareil photographique au moment de l'acquisition de l'image i. i peut prendre les valeurs g et d pour photographie gauche et photographie droite (fig. 1.7).
$R_i\,;T_i$	Matrices de rotation et de translation permettant de passer du repère terrain au repère lié à l'appareil photographique pour l'image i. On utilisera les indices d et g pour désigner les images de droite et de gauche(fig. 1.7).
$X_{pp1}\,;Y_{pp1}$	Coordonnées pixels du point principal de l'image 1(fig. 1.7).
$\omega_i\,;\phi_i\,;\kappa_i$	Angles de rotation équivalents à la matrice R_i. ω est une rotation d'axe X, ϕ d'axe Y et κ d'axe Z.

TAB. 1.1 – Lexique de photogrammétrie

1.2.2 Principe de la mesure de la position 3D d'un point

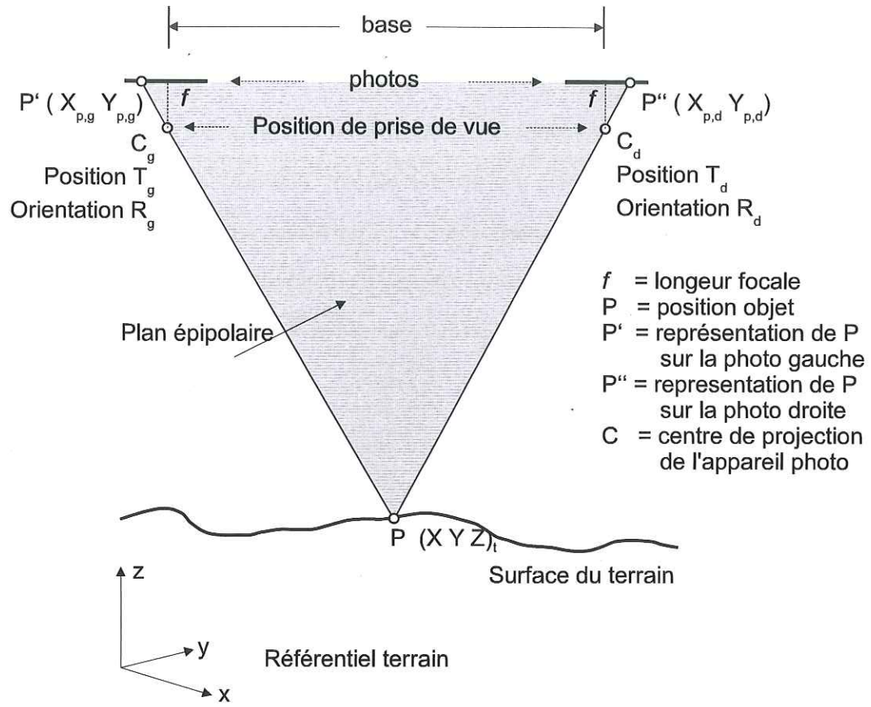


FIG. 1.7 – Calcul de la position d'un point par photogrammétrie.
Le point P se projette sur les deux photographie en P' etP''. Le calcul de la position des droites passant par les centre optiques $C_g$et C_d de l'appareil photographique dans chacune des positions de prise de vue permet de déterminer la position terrain de P. D'après Linder (2006)

Pour déterminer la position d'un point visible sur les deux photographies, il faut d'abord déterminer la position de ce point sur les images prises depuis deux points de vue différents. En vision monoscopique, la précision sur la coordonnée image du point que l'on souhaite mesurer est de ± 1 pixel. En pointé stéréoscopique on parvient à avoir une précision de $\pm\frac{1}{4}$ pixel [Thom 2002]. À partir de la position du point considéré sur l'image 1, on va calculer l'équation de la droite de l'espace qui se projette sur ce pixel. On procède de même pour l'image 2 (fig. 1.7). La position du point recherché est, en

théorie, l'intersection de ces deux droites.

Dans la pratique, si le pointé est fait sans asservissement sur l'épipolaire, et que l'annulation de la parallaxe n'est pas parfaite, les deux droites de la figure 1.7 ne sont pas sécantes. L'objet pointé peut alors être considéré comme étant situé au point qui minimise la distance entre les deux droites. La plupart des dispositifs (numériques comme mécaniques) imposent une annulation de la parallaxe. Le point de l'image 2 étant recherché uniquement sur l'épipolaire théorique au point de l'image 1. Dans ce cas de figure les deux droites sont nécessairement sécantes.

Le calcul de la position terrain (X_t, Y_t, Z_t) d'un objet P dont les images sur les deux photographies ont pour coordonnées (X_{p1}, Y_{p1}) et (X_{p2}, Y_{p2}) est régi par la suite d'équation ci dessous. Tout d'abord on prend en compte l'orientation interne (se reporter au lexique donné dans le tableau 1.1 pour la signification des différentes variables :

$$\begin{pmatrix} X_{mm} \\ Y_{mm} \end{pmatrix} = R_{interne}^{-1} \begin{pmatrix} X_p \\ Y_p \end{pmatrix} - T_{interne} \tag{1.1}$$

on tient compte de la position du point principal et de la distorsion :

$$x = X_{mm} - X_{pp}\left(-f^x\left(x, y\right)\right)$$
$$y = Y_{mm} - Y_{pp}\left(-f^y\left(x, y\right)\right)$$

Le détail de la fonction de distorsion f est donné plus loin dans ce paragraphe.

On calcule ensuite la position de l'objet sur le terrain dans le repère lié à l'appareil

photographique (axe Z parallèle à l'axe optique, Z positif vers l'arrière de l'appareil) :

$$\begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} = \lambda_1 \times \begin{pmatrix} x_1 \\ y_1 \\ -f \end{pmatrix} \quad \begin{pmatrix} X_2 \\ Y_2 \\ Z_2 \end{pmatrix} = \lambda_2 \times \begin{pmatrix} x_2 \\ y_2 \\ -f \end{pmatrix}$$

On en déduit la position de l'objet dans le repère terrain, à une inconnue λ_i près, d'après chaque équation.

$$\begin{pmatrix} X_{t,i} \\ Y_{t,i} \\ Z_{t,i} \end{pmatrix} = R_i^{-1} \times \begin{pmatrix} \lambda_i x_i \\ \lambda_i y_i \\ -\lambda_i f \end{pmatrix} - T_i$$

La position du point mesuré ne dépendant pas de l'image servant à le mesurer, on a les égalités suivantes : $X_{t,1} = X_{t,2} = X_t$; $Y_{t,1} = Y_{t,2} = Y_t$ et $Z_{t,1} = Z_{t,2} = Z_t$ qui équivalent au système de trois équations suivant :

$$R_1^{-1} \times \begin{pmatrix} \lambda_1 x_1 \\ \lambda_1 y_1 \\ -\lambda_1 f \end{pmatrix} - T_1 = R_2^{-1} \times \begin{pmatrix} \lambda_2 x_2 \\ \lambda_2 y_2 \\ -\lambda_2 f \end{pmatrix} - T_2 \tag{1.2}$$

Le système (1.2)ne possède que deux inconnues λ_1 et λ_2. Si la parallaxe est correctement annulée, le système est dégénéré et admet une unique solution. Sinon le système n'admet pas de solution. Dans ce cas la position généralement retenue est le point (X_t, Y_t, Z_t) solution du système ci-dessous et réalisant le minimum de $\epsilon_x^2 + \epsilon_y^2 + \epsilon_z^2$.

$$X_t(\lambda_1) + \epsilon_x = X_t = X_t(\lambda_2) - \epsilon_x$$
$$Y_t(\lambda_1) + \epsilon_y = Y_t = Y_t(\lambda_2) - \epsilon_y$$
$$Z_t(\lambda_1) + \epsilon_z = Z_t = Z_t(\lambda_2) - \epsilon_z$$

de la même manière, quand plus de deux images sont utilisées pour déterminer la

position d'un point, aucune valeur de $\lambda_1 \cdots \lambda n$ ne permettent de vérifier en même temps l'équation (1.2) pour toutes les valeurs de i. Il faut alors choisir le point (X_t, Y_t, Z_t) le plus proche de toutes les droites.

Pour calculer la position sur la photographie d'un point dont les coordonnées terrains sont connues, le principe est d'utiliser la même série d'équation dans l'autre sens, mais certaines équations se trouvent simplifiées par l'absence du paramètre λ. Tout d'abord il faut calculer la position dans le repère lié à l'appareil photographique au moment de la prise de l'image i (axe Z parallèle à l'axe optique, Z positif vers l'arrière de l'appareil) :

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = R_i \times \begin{pmatrix} X_t \\ Y_t \\ Z_t \end{pmatrix} + T_i \tag{1.3}$$

Ensuite on calcule la projection du point sur le plan focal :

$$x = -f \cdot \frac{X_i}{Z_i}$$
$$y = -f \cdot \frac{Y_i}{Z_i}$$

On tient compte de la position du point principal et de la distorsion :

$$X_{mm} = x + X_{pp}\,(+f^x\,(x, y)) \tag{1.4}$$

$$Y_{mm} = y + Y_{pp}\,(+f^y\,(x, y)) \tag{1.5}$$

Enfin on prend en compte l'orientation interne :

$$\begin{pmatrix} X_p \\ Y_p \end{pmatrix} = R_{interne} \begin{pmatrix} X_{mm} \\ Y_{mm} \end{pmatrix} + T_{interne} \tag{1.6}$$

La prise en compte de l'orientation interne (eq. (1.1) et (1.6)) est nécessaire avec les clichés argentiques parce que la pellicule n'est jamais placée exactement au même endroit dans l'appareil (cf. 1.2.4). Elle peut être omise pour les appareils numériques si on exprime directement en pixel la focale f, la distorsion et les coordonnées du point focal. Dans ce cas les équations (1.4) et (1.5) donnent directement X_p et Y_p en pixel.

1.2.3 Distorsion

Les fonctions $f^x\,(x, y)$ et $f^y\,(x, y))$ permettent de corriger les effets dus à l'imperfection de la lentille. Plusieurs modèles de distorsion existent. Le plus précis est la donnée de la valeur de la distorsion en un grand nombre de points de la photographie, utilisable dans le logiciel LISA [Linder, 2006], mais c'est aussi le moins pratique à utiliser. Dans les autres modèles, la distortion est décomposée en une composante radiale (eq. (1.7) et (1.8)), une composante tangentielle (eq. (1.9) et (1.10)), une composante de différence d'échelle entre l'axe x et l'axe y (eq. (1.11)) et une composante mesurant l'écart entre 90 degrés et l'angle formé entre l'axe X et l'axe Y (eq. (1.12)).

$$R_x = (x - X_{pp}) \cdot \sum_{i=1}^{n} k_i r^{2i} \text{ avec } r = \sqrt{(x - X_{pp})^2 + (y - Y_{pp})^2} \tag{1.7}$$

$$R_y = (y - Y_{pp}) \cdot \sum_{i=1}^{n} k_i r^{2i} \text{ avec } r = \sqrt{(x - X_{pp})^2 + (y - Y_{pp})^2} \tag{1.8}$$

$$T_x = P_1 \left(r^2 + 2\,(x - X_{pp})^2\right) + 2P_2\,(x - X_{pp})\,(y - Y_{pp}) \tag{1.9}$$

$$T_y = P_2 \left(r^2 + 2\,(y - Y_{pp})^2\right) + 2P_1\,(x - X_{pp})\,(y - Y_{pp}) \tag{1.10}$$

$$a_x = A\,(x - X_{pp}) \tag{1.11}$$

$$b_x = B\,(y - Y_{pp}) \tag{1.12}$$

Les paramètres k_i, P_1, P_2, A et B du modèle sont déterminés lors de la calibration de l'appareil photographique. Les fonctions de distorsion obtenus sont :

$$f^x(x,y) = R_x + T_x + a_x + b_x$$

$$f^y(x,y) = R_y + T_y$$

Les composantes de différence d'échelle et de non orthogonalité sont négligeables dans la plupart des cas pratiques. Elles ne sont requises que dans le cas d'objectifs de très mauvaise qualité. La composante tangentielle est aussi négligeable sur les objectifs de bonne qualité, sauf pour les objectifs à décentrement. On se contente donc la plupart du temps de seulement prendre en compte la distorsion radiale. Comme son nom l'indique elle ne dépend que de la distance avec le point focal et s'exerce en direction du point focal (vers lui ou en sens inverse). Elle peut donc être exprimée sous la forme $R_r = r \cdot D_r$. Les différents modèles varient sur la fonction la plus adaptée pour ajuster la distorsion radiale. La fonction la plus couramment utilisée est le polynôme des équations (1.7) et (1.8), $D_r = \sum_{i=1}^{n} k_i \cdot r^{2i}$. Généralement les trois premiers termes sont suffisants pour que la distorsion résiduelle soit inférieure au pixel. Mais d'autres modèles de distorsion sont parfois utilisés, comme par exemple $D_r = \sum_{i=1}^{n} K_i \cdot r^{i-1}$ ou $D_r = \sum_{i=1}^{n} A_i \cdot r\,(r^{2i} - r_0^{2i})$.

1.2.4 Calcul de l'orientation des images

Pour la mesure de la position d'un point (cf. 1.2.2), nous avons besoin de connaître la position et l'orientation de l'appareil de prise de vue au moment de la prise de vue. Cette opération peut être décomposée en trois étape : L'orientation interne, l'orientation relative et l'orientation externe absolue.

L'orientation interne revient à calculer la position et l'orientation du film photographique dans le plan focal de l'appareil photographique. Afin de pouvoir réaliser cette opération, les appareils argentiques utilisés pour la photogrammétrie comportent des repères de fond de chambre qui sont placés de manière fixe à l'intérieur de l'appareil photographique et sont matérialisés sur chaque cliché (Fig 2.3). On obtient ainsi sur les tirages 4 ou 8 marques précises (généralement des croix) situées dans les coins ou au milieu des cotés. Sur les appareils ne disposant pas de marques de fond de chambre, il est possible de se baser sur la position des bords de l'image et utiliser les intersections entre les bords comme des marque de fond de chambre, mais la précision de l'orientation interne est alors moins bonne. Connaissant les coordonnées (X^i_{mm}, Y^i_{mm}) de chacune des croix dans le plan focal en mm et leurs coordonnées (X^i_p, Y^i_p) sur l'image en pixel, on calcule les deux matrices $R_{interne}$ et $T_{interne}$ telles que

$$\begin{pmatrix} X_{p,i} \\ Y_{p,i} \end{pmatrix} = R_{interne} \begin{pmatrix} X_{mm,i} \\ Y_{mm,i} \end{pmatrix} + T_{interne}$$

Le système est redondant dès que l'on mesure plus de trois marques de fond de chambre, il faut donc minimiser les erreurs. On résout alors le système

$$\begin{pmatrix} X_{p,i} \\ Y_{p,i} \end{pmatrix} = R_{interne} \begin{pmatrix} X_{mm,i} \\ Y_{mm,i} \end{pmatrix} + T_{interne} + \begin{pmatrix} \epsilon_{x,i} \\ \epsilon_{y,i} \end{pmatrix}$$

en minimisant $\sum_i \epsilon_i^2$. Cette étape n'est pas indispensable avec les appareils numériques parce que le capteur est toujours à la même place. Les étapes d'orientation relative et absolue sont alors réalisées en travaillant sur les coordonnées pixels, à condition que les paramètres de calibration décrits au paragraphe 1.2.6 soient tous exprimés en pixel.

L'orientation relative est le calcul de l'orientation et de la position d'une image par rapport à l'autre, indépendamment de tout point d'appui terrain. On choisit en général d'exprimer l'orientation et la position de l'image droite en fonction de l'image gauche.On calcule alors les matrices R_d et T_d en supposant $R_g = I$ (Image gauche

parfaitement dans le plan X, Y, axes X_t, Y_t et X_g, Y_g confondus) et $T_g = \vec{0}$. En l'absence de tout appui terrain sur les photographies, T_d ne sera défini qu'à un facteur d'échelle près, c'est à dire que si T_d est une solution, $\lambda \cdot T_d$ en est une autre. On impose en général $\|T_d\| = 1$. D'autres conditions peuvent être imposées en lieu et place de celles indiqués ci-dessus, comme par exemple $T_g = \vec{0}$ et $T_d = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ et « la composante selon Y de l'axe optique de l'image gauche est nul ».

Les matrices de rotation R peuvent être exprimés à partir des angles ω, ϕ, κ. Elle correspondent donc chacune à trois inconnues scalaires. Le problème comporte donc douze inconnues au départ (ω_i, ϕ_i, κ_i, $T_{x,i}$, $T_{y,i}$, $T_{z,i}$ avec $i = d$ et $i = g$) dont sept peuvent être imposées (par exemple les paramètre de l'image gauche et la norme de T_d). Chaque point homologue P ajoute trois inconnus (les coordonnées terrain du point $X_{t,P}$, $Y_{t,P}$, $Z_{t,P}$) et quatre équations liant les coordonnées terrain aux quatre coordonnées image et aux paramètres d'orientation. Ces équations dérivent des équations (1.3) à (1.6). Sauf cas particuliers (trois points alignés ou quatre points coplanaires), il suffit donc d'identifier cinq points homologues entre les deux images pour déterminer les inconnues restantes. Si plus de 5 points sont mesurés, une minimisation des erreurs résiduelles est effectuée pour établir l'orientation relative.

L'orientation absolue ou orientation externe est le calcul des positions et orientations des deux photographies par rapport au référentiel terrain. Ce calcul nécessite de connaitre les positions terrain de points d'appui identifiables sur les photographies. Une photographie isolée peut être orientée de cette manière. Dans ce cas le nombre d'inconnues à déterminer est de six (les deux matrices R et T). Chaque point terrain mesuré ajoute deux équations liant les coordonnées terrains et photos entre elles. Trois points connus en X, Y, Z sont donc nécessaires pour déterminer les deux matrices R

et T. Dans le cas d'un couple stéréoscopique, il est possible de réaliser cette orientation avec moins d'éléments connus. Après l'étape d'orientation relative, il ne reste plus qu'à déterminer les sept inconnues qui avaient été imposées précédemment. Tout point connu en X ou en Y ou en Z ajoute une contrainte en diminuant d'un le nombre d'inconnues de l'orientation relative. Le nombre de points d'appui minimum est donc de trois points connus en (X, Y, Z) (9 contraintes), ou de deux points connus en (X, Y) plus trois points connus en Z uniquement (7 contraintes), ou encore deux points connus en (X, Y) et un connu en (X, Y, Z).

Les orientations externes relative et absolue peuvent être calculées en une seule fois, les points d'appui permettant de mieux contraindre l'orientation relative. Les erreurs sur les points homologues et sur la position des points d'appui sont alors minimisées globalement, au lieu de minimiser les erreurs lors de l'orientation relative, et d'imposer l'orientation relative lors de l'orientation absolue. La possibilité de calculer une orientation relative séparément reste utile, par exemple lorsqu'on souhaite exploiter un couple d'images alors que l'on ne dispose pas ou pas encore de point d'appui. Dans ce cas les points mesurés sur le modèle photogrammétrique sont exprimés dans le repère terrain imposé par le choix des sept paramètres libres au moments de l'orientation interne. Pour pouvoir utiliser ce modèle il est presque toujours nécessaire de disposer d'un élément de mise à l'échelle et d'un moyen de déterminer l'axe vertical.

1.2.5 Aerotriangulation

Il arrive que l'objet à étudier soit trop proche pour être entièrement visible sur un seul couple stéréoscopique, soit parce qu'on ne dispose pas de point de vue suffisamment lointain soit parce que les besoins en résolution obligent à photographier l'objet de près. On utilise alors plusieurs couples successifs, ou l'image droite du premier couple devient

l'image gauche du second couple et ainsi de suite. Si le premier couple possède assez de points d'appui pour que son orientation soit déterminée, R et T sont déterminés pour l'image gauche du deuxième couple. Cela fixe six des sept paramètres laissés libres par l'orientation relative. Il suffit alors d'un seul point d'appui pour déterminer l'orientation absolue de l'image droite (du second couple). Si une seule des coordonnées (X, Y, Z) de ce point est connu il doit être visible sur les deux images droite et gauche, si au moins deux sont connus il peut n'être visible que sur l'image droite. En poursuivant la méthode, il suffit d'un seul point connu en (X, Y, Z) sur chaque image pour déterminer la position de toutes les images.

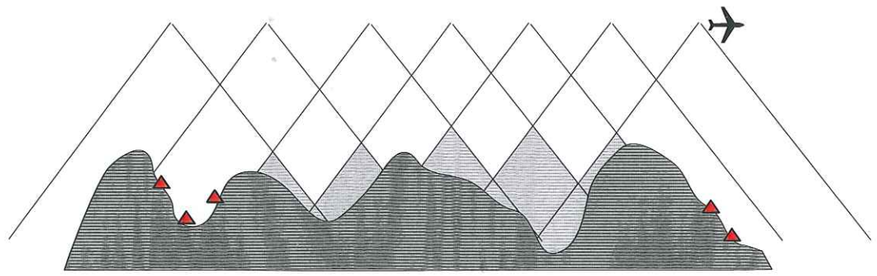


Fig. 1.8 – Principe de l'aérotriangulation.
La présence de points de liaisons dans les zones couvertes par trois photographies (zones en gris clair) permet de calculer l'orientation absolue de toutes les photographies de la bande à partir de points d'appuis (triangle rouge) visibles uniquement depuis les photographies situées aux extrémités de la bande

Si le recouvrement entre les images est supérieur à 50%, il existe des points visibles sur trois images succesives $i, i+1, i+2$. De tels points sont appellés points de liaison. Une fois connues les orientations absolues de images i et $i+1$, la position des points de liaison peut être calculée en utilisant les équations vues en 1.2.2. La position de l'image $i+2$ est alors déterminée en utilisant ce point de liaison sans disposer de réels points d'appuis visible sur cette image (fig. 1.8). C'est le principe de l'aérotriangulation.

Dans la pratique les erreurs se propagent fortement si l'on utilise un seul point de

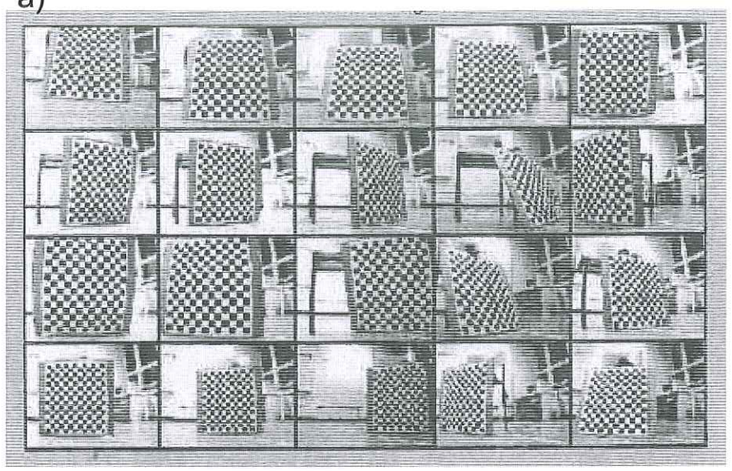
liaison. Pour limiter la propagation des erreurs d'orientations, on s'efforce en pratique de disposer de bien plus de points de liaisons que le seul nombrestrictement nécessaire et de bien plus de points homologues que les 5 nécessaires à l'orientation relative. Le système d'équation est alors fortement redondant, et le calcul de l'ensemble des orientations de l'ensemble des images est fait en même temps, de manière à minimiser globalement les erreurs. Avec une détermination manuelle on utilise en général une quinzaine de points homologues par image, cette valeur passe à une centaine lorsque la détermination des points homologues et des points de liaison est automatique, afin de compenser leur moins bonne répartition dans l'image et les risques de fausse corrélation [Jung *et al.*, 2002].

1.2.6 Calibration de l'appareil photographique

Le présent paragraphe se limite aux aspects de la calibrations utilisable aussi bien pour les appareils argentique et numérique. Pour les appareils argentiques une partie des paramètres de calibration peut être déterminée par des mesure physiques réalisées sur le boitier ouvert, en l'absence de pellicule. Cet aspect de la calibration ne sera pas abordé ici.

La calibration de l'appareil photographique est un préalable indispensable à l'exploitation de clichés photogrammétriques. C'est le calcul des paramètres intrinsèques à l'appareil que nous avons utilisés dans les sections précédentes : distance focale, position du point focal et paramètres de distorsion, position des marques de fond de chambre pour les appareils argentiques. Ces paramètres sont trop sensibles aux petits défauts de la lentille pour pouvoir être considérés comme identiques sur tous les appareils d'un modèle donné. Ils doivent donc être mesurés pour chaque appareil. Ces paramètres étant liés au montage des pièces de l'appareil, ils sont supposés ne pas varier au cours du

a)



b)

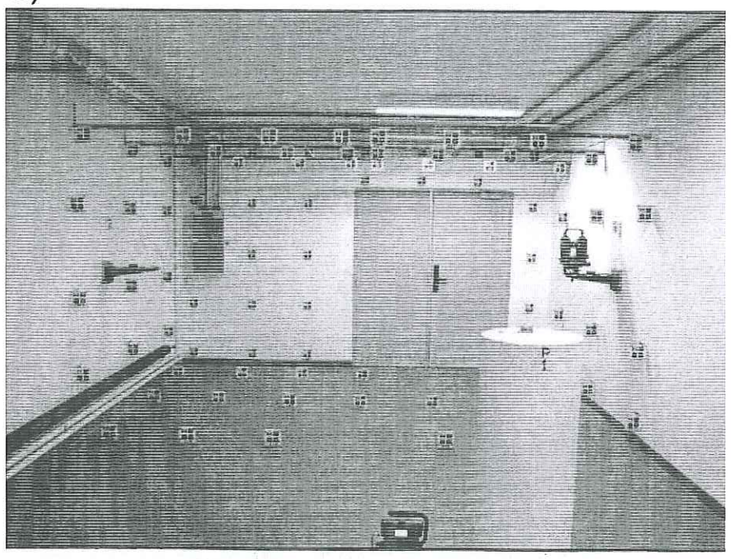


Fig. 1.9 – Ploygone et mire de calibration
a) Calibration par mire plane, il est nécessaire d'être proche de la mire, donc la distance focale n'est pas à l'infini.
b) Exemple de polygone de calibration, les nombreuses cibles sont fixes et leurs positions mesurées précisément.

temps. Cependant, une fatigue mécanique est envisageable et les appareils très utilisés doivent être recalibrés à intervalle régulier.

Deux principaux protocoles de calibration d'appareil photos sont utilisés : la mire et le polygone de calibration (fig. 1.9). La mire de calibration est le plus simple à mettre en œuvre [Zhang 1999]. La mire est constituée d'un damier dont la dimension des cases noires et blanche est connue et régulière. Les imprimantes laser sont suffisamment précise pour produire de telles mires de dimension allant de A5 à A3. Avec une mire plane, plusieurs prises de vue sont nécessaires au calcul des paramètres intrinsèques ; si l'on dispose d'une mire formée de deux plans côte à côte et que l'angle entre les plans est connu, une seule prise de vue suffit. Le calcul des paramètres se fait en résolvant les équations (1.3) à (1.5) pour chaque coin d'un carreau de la mire et chaque prise de vue. Les inconnues à déterminer sont les paramètres intrinsèques (c'est à dire de calibration) et les paramètres d'orientation, les positions terrain et image des coins des carreaux étant connues. Le système d'équation étant non linéaire, la résolution se fait par itérations successives. Étant donné le grand nombre de point d'appui sur la mire (224 pour la mire présentée sur la figure 1.9), la mesure est relativement précise même avec peu d'images : dès 3 images les paramètres sont déterminés avec une incertitude inférieure à 0.3% pour f et une incertitude de 1 pixel sur la position du point principal [Zhang 1999].

La principale limite de cette méthode est la petite taille des mires. En effet, les paramètres intrinsèques, notamment f, dépendent du réglage de la distance de mise au point. Comme on souhaite en général étudier des objets lointains, l'objectif doit être réglé sur l'infini et les paramètres intrinsèques déterminés dans cette position. Les objets proches sont alors flous. On doit s'éloigner de la mire pour que celle-ci soit nette. La mire ne couvre alors plus une part suffisante du champ photographié et les carreaux deviennent de dimension inférieure au pixel.

La distance minimum au delà de laquelle on peut considérer un objet comme net est appelée distance hyperfocale H. C'est la distance pour laquelle un objet ponctuel aura l'apparence sur l'image d'une tache d'un diamètre supérieur à une tolérance de netteté fixée . H est donné par $H = \frac{f^2}{o \cdot c}$ où f est la focale, o l'ouverture et c la dimension de la tolérance de netteté. Pour l'appareil numérique que nous avons eu à calibrer, f=24mm, $o = 1$ pour ne pas allonger le temps de pose, les photographies de falaise étant prises depuis hélicoptère, $c = 8\mu m$ (c'est la dimension d'un pixel) soit H=3m. Les mires obtenus par impression laser ne sont plus d'une taille suffisante à cette distance. Des mires de plus grande dimension sont plus difficile à produire en gardant la même précision.

Le polygone de calibration est un dispositif comme celui de la figure 1.9. Les cibles doivent être bien réparties sur les trois coins du dispositif afin que celui-ci soit le plus efficace possible. Le nombre de cibles étant plus réduit que pour la mire plane, typiquement quelques dizaines, il faut prendre plus de prises de vue différentes que pour une calibration par mire afin de déterminer avec précision la distorsion sur toutes les parties de l'image. Les paramètres intrinsèques sont calculés de la même manière que précédemment, ce qui signifie que les positions des cibles doivent être connues avec une grande précision. Ce dispositif peut être installé sur une façade en coin de bâtiment, ce qui permet d'avoir un recul bien plus grand que dans le cas des mires. L'ensemble de la zone photographiée est plus complexe que dans le cas d'une mire plane, et l'identification automatique des cibles n'est pas toujours possible. L'appareil photo décrit ci-dessus a été calibré en utilisant le polygone de calibration du politecnico de Turin.

1.2.7 Avantages comparatifs de l'argentique et du numérique

Les appareils photographiques argentiques et numériques ont des caractéristiques assez différentes qui ont une influence notable sur leur utilisation en photogrammétrie. Les principales différences sont notés ci-dessous :

- Les différentes technologies numériques : Deux types de capteurs différents sont utilisés dans les appareils numériques, les capteurs CCD et les capteurs CMOS. Dans les capteurs CMOS, chaque photosite est entouré par plusieurs transistors qui réalisent la conversion entre signal analogique et signal numérique (fig. 1.10), alors que dans les capteurs CCD cette conversion est réalisé par un seul convertisseur pour l'ensemble du capteur.

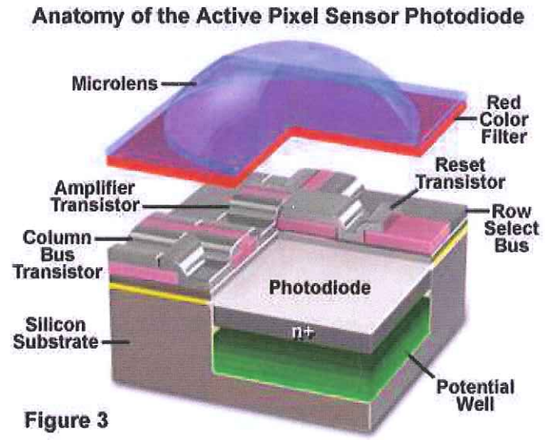


FIG. 1.10 – Schéma d'un pixel d'un capteur CMOS.
Seule la photodiode capte la lumière, la proportion de lumière captée est plus faible que sur un capteur CCD, ce qui augmente le rapport signal sur bruit. Le signal est converti en signal numérique par les transistors au niveau du photosite, et non par un convertisseur unique pour l'ensemble du capteur

Chaque pixel du capteur CMOS peut donc être lu indépendamment tandis que l'ensemble du capteur CCD doit être lu à chaque acquisition. En contrepartie la totalité d'un pixel reçoit la lumière sur un capteur CCD alors que seule la partie

photodiode du pixel la reçoit dans un capteur CMOS. Les capteurs CCD sont de ce fait moins sensible au bruit que les capteurs CMOS. Ils consomment par contre cent fois plus d'énergie que les capteurs CMOS et sont nettement plus cher, car les capteurs CMOS peuvent être fabriqués sur des lignes de production silicone classiques.

- Prix : Les appareils numériques réflex destinés aux photographes amateurs ou semi-professionnel sont utilisables pour de la photogrammétrie dès que certaines conditions sont remplies : il faut que la focale et la mise au point puissent être fixés, c'est à dire que l'appareil ne soit pas équipé d'un dispositif autofocus non débrayable, ne possède pas de stabilisateur d'image optique et que l'objectif, grand angle, soit de bonne qualité. On trouve ainsi des appareils de bonne qualité avec des résolutions de 12 millions de pixels à moins de 5000 euros. Il est possible d'utiliser en photogrammétrie des réflex argentiques standards, même si la mesure de l'orientation interne en est plus compliqué que pour les chambres métriques dédié à la photogrammétrie. Au coût de l'appareil argentique, actuellement toujours inférieur à celui du numérique équivalent, il faut ajouter celui du développement des négatifs et du tirage et celui de leur numérisation. Le surcoût de la solution numérique peut donc être rapidement amorti.
- Résolution : L'ouverture angulaire utilisée étant à peu près constante, la résolution angulaire est directement fonction de la dimension de l'image. La dimension de l'image pour les appareils numériques est fixée par l'appareil photographique lui même. À l'heure actuelle les meilleurs appareils ont une résolution de l'ordre de 40 millions de pixel. La résolution dans le cas d'un appareil argentique dépend directement de la taille du négatif. Ceux-ci peuvent être scannés à une résolution de 8 μm ce qui correspond à une dimension de 14 millions de pixel pour un réflex 24×36mm.
- Disponibilité : le cliché numérique est immédiatement disponible, tandis que le fichier argentique doit être développé puis scannerisé pour être utilisable. Ce délai n'est pas forcément très long, mais doit être pris en considération lorsque l'intervention est faite en urgence.

- Le support : le capteur numérique est fixe par rapport à l'objectif. Le film argentique ne l'est pas. La phase d'orientation interne, nécessaire pour prendre en compte la position du film argentique au moment de la prise de vue, peut être évitée avec les appareils numériques. De plus le film argentique est déformable et la présence de poussière entre celui-ci et le boitier de l'appareil photographique peut créer localement des distorsion non modélisables et difficilement décelables [Egels, 2002b].
- Dynamique de niveau de gris : les appareils numériques sont capables de différencier jusqu'à 2^{12} niveaux de gris tandis qu'on obtient 2^6 niveaux de gris différents à partir de photographies argentiques dans les meilleures conditions [Kasser, 2002]. Ceci s'explique entre autre par une réponse linéaire des capteurs numériques au nombre de photons reçus, alors que le film argentique répond de manière logarithmique. La meilleure dynamique des appareils numériques permet de percevoir des contraste de couleur dans des zones semblant homogènes sur les photographies argentiques.
- Bruit : la principale source de bruit est le bruit de photon, lié à la nature corpusculaire de la lumière. Ce bruit est proportionnel au carré du nombre de photons détectés. Pour un appareil numérique avec une dimension du pixel de 9μm, le nombre de photons détectés est de l'ordre de 20 000, donnant un rapport signal sur bruit d'environ 350. Pour un cliché optique le nombre de photon détecté est proprotionnel au carré du rapport entre la surface du pixel et la surface des grains d'argent. Pour un pixel de 20μm et un grain de 2μm on obtient un rapport signal sur bruit $r \approx 17$ considérablement plus faible [Thom, 2002].
- Couleur : les films couleurs argentiques sont composés de trois couches diffé-

rentes. Chaque couche est, en théorie, sensible à une seule des trois longueurs d'onde fondamentale correspondant aux trois couleurs (rouge, vert, bleu). Tout point du film enregistre donc séparément les intensités de la couche bleue, de la couche verte et de la couche rouge. Les capteurs numériques possèdent un filtre coloré sur chaque photosite. On a globalement seulement $\frac{1}{4}$ des photosites sensible au bleu, $\frac{1}{2}$ des photosites sensible au vert et $\frac{1}{4}$ des photosites sensible au rouge. La couleur de chaque pixel de l'image finale est réalisée par interpolation des couleurs sur les quatre photosites qui encadrent le pixel. Seuls des systèmes tels que les capteurs tri-CCD, présent uniquement sur des appareils bien spécifiques, permettent de capter l'intensité de chacune des trois couleurs séparément. Si la couleur en chaque point de l'image est interpolée sur les appareils numériques alors qu'elle est enregistrée partout sur les appareils argentiques, ceci est compensé par le fait que les filtres séparent plus efficacement les longueurs d'ondes que ne le font les différentes couches des films couleurs.

Dans le cadre d'une utilisation avec un appareil photographique grand public ou semi-professionnel, l'argentique n'est avantageux que sur l'investissement initial, à condition que le nombre d'utilisation reste suffisamment faible pour que le surcoût du numérique ne soit pas compensé par les coût de développement, tirage et scan. Pour des applications où une très grande résolution est nécessaire, des chambres argentiques grand format(13*18cm) comme des dispositifs numérique de très haute résolution existent. Le coût de tels dispositifs restreint leur utilisation à des photogrammètres professionnels, leur comparaison ne sera donc pas faite ici même si les remarques faites plus haut restent pour la plupart valable. Notons tout de même que le faible prix actuel des chambres photogrammétrique moyen format d'occasion rend cette alternative envisageable.

1.2.8 Corrélation automatique

Pour faciliter le pointé des détails caractéristiques, les logiciels de photogrammétrie proposent souvent des ajustement par corrélation, ponctuels ou permanents (asservissement au sol). L'ajustement est fait en cherchant la valeur de Z_t donnant lieu à la meilleure corrélation (c'est à dire à des images étant le plus semblable l'une de l'autre) pour les valeurs de X_t et Y_t pointé par l'utilisateur. Le choix de l'axe Z comme axe d'ajustement lors de la corrélation est naturel lors de la photogrammétrie aérienne, mais n'est pas judicieux pour la photogrammétrie terrestre. Malheureusement la quasi-totalité des logiciels de photogrammétrie ont été conçus pour la photogrammétrie aérienne, et l'utilisateur est souvent obligé de définir un repère terrain avec l'axe Z sub-parallèle à l'axe optique des photographies pour pouvoir tirer parti des ajustements par corrélation.

Cette corrélation automatique peut aussi être effectuée pour tous les pixels de l'image, permettant ainsi d'obtenir rapidement un MNT de la zone photographié. Pour déterminer si deux points sont homologues ou non, une fonction de corrélation va déterminer si les pixels et leurs voisinages sont similaires ou non. Cette fonction valant 1 si les deux images sont identiques et -1 si elles sont très différente, la corrélation entre deux images aléatoires étant de 0.

L'expression la plus simple de cette fonction de corrélation est

$$\rho(x_1, y_1, x_2, y_2) = \frac{\sum_{\substack{i=-n..n \\ j=-n..n}} g^{(1)}_{x_1+i,y_1+j} \cdot g^{(2)}_{x_2+i,y_2+j}}{\sqrt{\sum_{\substack{i=-n..n \\ j=-n..n}} {g^{(1)}_{x_1+i,y_1+j}}^2 \cdot \sum_{\substack{i=-n..n \\ j=-n..n}} {g^{(2)}_{x_2+i,y_2+j}}^2}} \tag{1.13}$$

où $g^{(i)}_{a,b}$ indique le niveau radiométrique du pixel (a, b) de l'image i, les pixels dont on veut calculer le coefficient de corrélation étant les pixels x_1, y_1 pour l'image 1 et x_2, y_2

pour l'image 2. Pour des images en couleur, la fonction de corrélation globale est :

$$\rho_{globale} = \rho_{r(rouge)} \times \rho_{v(vert)} \times \rho_{b(blue)}$$

Pour les images contenant n couches à différentes longueurs d'onde, on a :

$$\rho_{globale} = \prod_{i=1}^{n} \rho_{\lambda_i}$$

Lorsque l'orientation externe est connue, la recherche du pixel homologue à un pixel donné n'est pas faite sur toute l'image, mais seulement le long de l'épipolaire à ce pixel, le pixel répondant à la meilleure corrélation étant retenu. On considère que la corrélation est bonne si $\rho \geq 0.7$ pour une image en noir et blanc et $\rho \geq 0.7^3 = 0.35$ pour les images en couleur. En dessous de ces valeurs on considère que la corrélation n'est pas bonne et que le point homologue n'a pas pû être trouvé. Pour s'affranchir d'une différence de contraste, locale ou globale, entre les deux images, l'équation (1.14) utilise, à la place du niveau radiométrique absolu, l'écart avec $g_m^{(i)}$, le niveau radiométrique moyen à l'intérieur du carré de coté $2n+1$ centré sur x_i, y_i.

$$\rho = \frac{\sum_{\substack{i=-n..n \\ j=-n..n}} \left(g^{(1)}_{x_1+i,y_1+j} - g^{(1)}_m\right) \cdot \left(g^{(2)}_{x_2+i,y_2+j} - g^{(2)}_m\right)}{\sqrt{\sum_{\substack{i=-n..n \\ j=-n..n}} \left(g^{(1)}_{x_1+i,y_1+j} - g^{(1)}_m\right)^2 \cdot \sum_{\substack{i=-n..n \\ j=-n..n}} \left(g^{(2)}_{x_2+i,y_2+j} - g^{(2)}_m\right)^2}} \tag{1.14}$$

Le résultat de ces deux fonctions de corrélation est fortement dépendant de la valeur de n : si n est trop petit, des détails similaires à des endroits distincts seront bien corrélés, si au contraire n est trop grand, aucune corrélation ne sera trouvée. Pour palier ce problème, Dequal *et al.* (1996) introduisent une fenêtre de recherche de corrélation divisée en trois zones ayant chacune le même poids global (fig. 1.11). L'influence de la zone externe limite les risques de corréler des détails similaires à des endroits différents, tout en étant moins importante que la bonne concordance des détails au niveau local.

Les équations (1.13) et (1.14) supposent que la zone de l'objet étudié représentée par un carré sur l'image 1 est un carré de même dimension et même orientation sur l'image 2. Ce qui revient à supposer que :

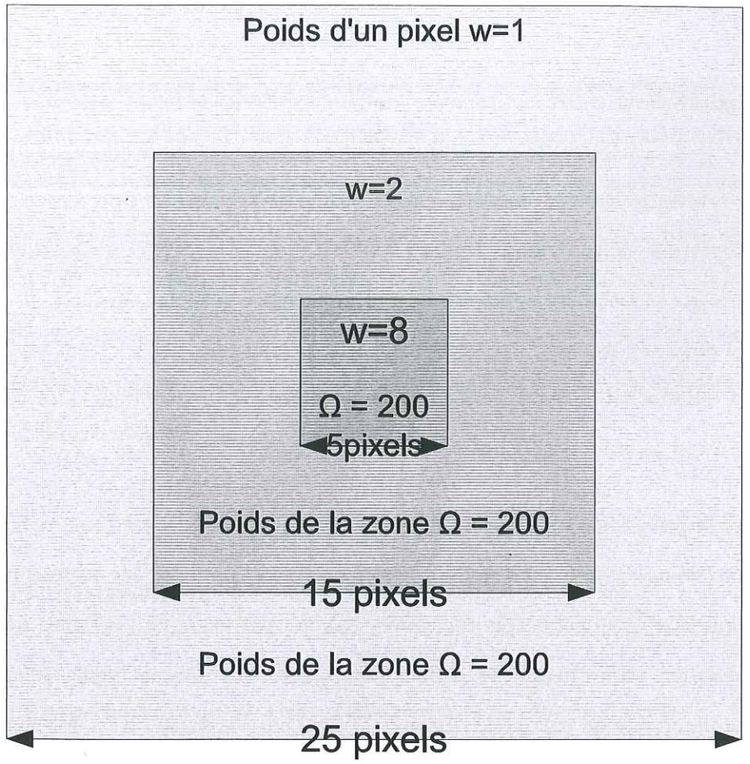


FIG. 1.11 – Fenêtre de corrélation pondérée.
Les pixels des trois zones ont un poids différent dans la fonction de corrélation, les trois zones ont le même poids global. D'après Dequal et al. *(1996).*

- Les images sont prises à la même distance et dans la même orientation
- Les points à l'intérieur de la fenêtre de recherche ont sensiblement la même altitude, c'est à dire que les positions des pixels de la fenêtre de recherche s'éloignent peu d'un plan perpendiculaire à l'axe optique de l'appareil.

Dans la pratique, ces conditions sont toujours vérifiées en photogrammétrie aérienne. En photogrammétrie terrestre ce n'est pas toujours le cas. Adapter cette formule pour tenir compte de différences d'orientation entre les deux photographies est possible puisque la forme que prend sur l'image 2 ce qui est un carré sur l'image 1 peut être calculé à priori. L'adaptation de la deuxième condition est plus problématique car non

prévisible à priori. Pour palier ce problème, on peut généraliser la méthode précédente en supposant que la fenêtre de corrélation, carrée sur l'image de droite, a subi une déformation arbitraire sur la zone de gauche. Il est dans ce cas exclu de rechercher le point homologue sur toute la longueur de l'épipolaire. La stratégie de recherche de la fenêtre homologue prend alors appui sur la continuité de l'objet photographié. En effet comme notre objet est continu, deux fenêtres contigües sur l'image 1 le sont aussi sur l'image 2. La corrélation se fait alors de proche en proche à partir de quelques germes de départ, souvent positionnés visuellement par l'opérateur humain.

Une autre approche pour l'identification des points homologue est l'approche dite par structure [Wang, 1998]. L'algorithme commence par identifier les formes caractéristiques présentes sur chaque photographie, comme par exemple les lignes de fort gradient de couleur, ou les région de couleur homogène. Puis il structure ces éléments en définissant les relations entre eux. Ensuite il met en correspondance les éléments de l'image 1 et ceux de l'image 2. Cette méthode à l'avantage de fonctionner même lorsque l'orientation relative des deux images n'est pas connue à priori. Elle est aussi nettement plus efficace en photogrammétrie convergente, où l'aspect d'un même détail peut être très différent entre les deux photographies.

La présence d'erreurs de corrélation est l'un des principaux problèmes de la corrélation automatique. Suivant la technique employée et la qualité du contraste sur les photographies, ces erreurs seront plus ou moins fréquentes. Un traitement du nuage de points résultant de la corrélation est de ce fait indispensable pour éliminer les points aberrants. Une visualisation stéréoscopique des points obtenus en superposition avec le couple de clichés permet de vérifier qu'il ne reste pas trop d'erreurs de corrélation : les points corrects apparaissent visuellement au contact de la paroi tandis que les points erronés apparaissent soit en avant soit en arrière de la paroi.

Contrairement à l'exploitation manuelle où une large base photogrammétrique permet une bonne perception du relief, la corrélation automatique s'accommode très bien d'une base courte. Le fait de raccourcir la base impliquerait une perte de précision sur la position terrain des points corrélés si la précision de la position image était identique, mais le raccourcissement de la base rend les deux images plus ressemblantes, permettant de gagner en précision de la position image des points homologues. La précision finale de la position terrain est donc à peu prêt constante et le risque d'erreur de corrélation est réduit.

1.3 Scannerisation laser

1.3.1 Fonctionnement des différents scanners laser

Le principe des scanners laser, ou scanners LIDAR (LIght Detection And Ranging), est de mesurer la distance entre l'émetteur d'un rayon laser et l'objet qui va rétrodiffuser ce rayon. Cette mesure est répétée rapidement un grand nombre de fois dans de nombreuses directions connues. Lors de l'impact d'un rayon lumineux avec une surface, la majeure partie de l'énergie est réfléchie selon la loi de réflection de Descartes [Descartes, 1637]. À cause de la granulosité à petite échelle de la surface réfléchissante, une part de cette énergie est diffusée dans toutes les directions, y compris dans la direction d'émission (fig. 1.12). L'intensité du rayon rétrodiffusé dépend de l'angle d'incidence du rayon incident et de l'état de surface à petite échelle du matériau sur lequel a lieu la réflection (rugosité locale notamment). Par la suite nous parlerons simplement de réflection pour évoquer ce phénomène de rétrodiffusion. Le faisceau laser est très peu divergent ($< 0.3mrad$), permettant généralement de considérer le réflecteur comme ponctuel. L'impulsion laser est brève et temporellement cohérente, permettant

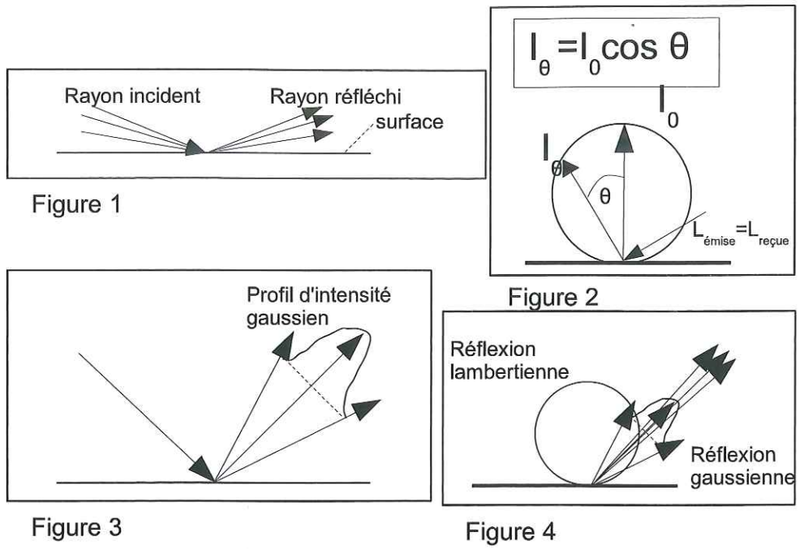


FIG. 1.12 – Principe de la rétrodiffusion.
Lors de la rencontre d'un rayon lumineux avec une interface, on observe un rayon réfléchi (1) et un rayon réfracté (lois de Descartes). On observe aussi une diffusion gaussienne autour du rayon réfléchi théorique (2) et une diffusion de Lambert (3). L'énergie réfléchie dans toutes les directions est lié à la superposition des trois phénomènes (4). [Stenberg, 1996]

de considérer l'émission et la réflection comme instantanées. Nous reviendrons sur ces hypothèses en 1.4. La direction d'émission du rayon laser change automatique après chaque impulsion afin d'émettre et de recevoir successivement dans de nombreuses directions de l'espace. Suivant le type d'appareil ce changement de la direction d'émission peut être réalisé simplement soit par un miroir situé sur le trajet du rayon laser, soit par une rotation de l'ensemble de l'appareil de mesure ; par exemple le laser Riegl LMS Z 420 (fig. 1.13) comporte un miroir tournant autour d'un axe horizontal (quand le laser est en position verticale, comme sur la figure 1.13), qui permet au rayon laser de balayer les angles θ compris entre $-40°$ et $+40°$, et l'ensemble de l'instrument tourne sur sa base afin que l'émission couvre tous les angles ϕ compris entre 0 et 360°.

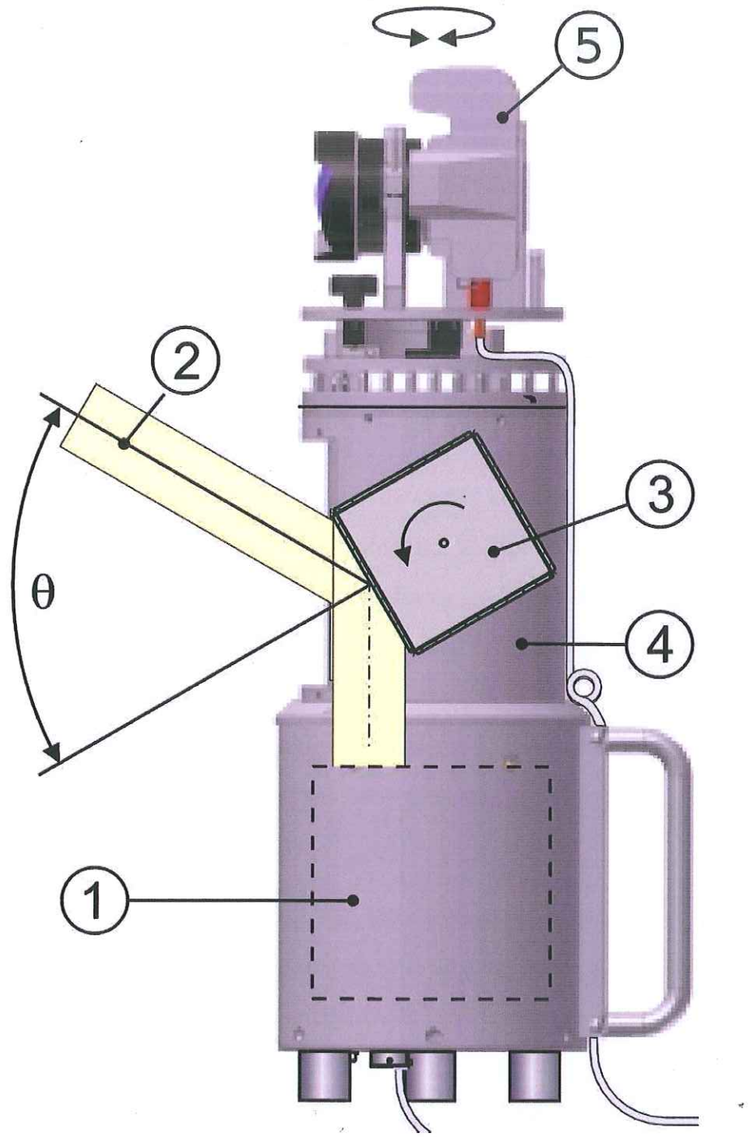


FIG. 1.13 – Schema du principe de fonctionnement du laser Riegl LMS Z 420.
1 : Émetteur et récepteur laser. 2 : Rayon laser émis selon un angle θ pouvant varier de $+40°$à $-40°$. 3 : Miroirs tournants assurant la variation de l'angle θ. 4 : L'ensemble de la tête optique tourne pour faire varier l'angle ϕ entre 0 et 360°. 5 : Appareil photographique. [rieglldatasheet,]

1.3.1.1 Laser à mesure de phase

De tous les principes de mesure de distance par laser, la mesure de phase donne le résultat le plus précis mais c'est aussi la méthode la plus lente. La mesure se fait en regardant comment la fréquence du signal laser modifie la phase du signal retour. Le laser émet un signal de longueur d'onde λ. Le signal parcourt une distance D jusqu'au réflecteur, puis revient dans l'autre sens. Il subit un déphasage de :

$$2\pi \cdot \frac{2D}{\lambda}$$

Si la longueur d'onde varie, la phase du signal reçu en retour varie également. En variant la longueur d'onde λ il est possible d'obtenir un déphasage identique à celui observé pour λ avec une autre longuer d'onde $(\lambda - \epsilon)$. Le déphasage etant défini modulo 2π, on a donc :

$$2\pi \cdot \frac{2D}{\lambda} = 2\pi \cdot \frac{2D}{\lambda - \epsilon}[2n\pi]$$

on en tire l'expression

$$D = \frac{\lambda\,(\lambda - \epsilon)}{2\epsilon}$$

La variation de la longueur d'onde du laser devant être mesurée précisément, la mesure d'un point prend, à cent mètres quelques dixièmes de secondes à quelques secondes. Il faut de plus que le point de mesure et l'objet soient immobiles, toute variation de distance pendant la mesure ayant un impact sur la phase observée, sauf si la variation est négligeable devant λ. Ce type de laser peut donc difficilement être utilisé pour acquérir la topographie complète d'un objet complexe nécessitant la connaissance de plusieurs millions de points. C'est par contre le principe qui est utilisé par les stations totales ou les distancemètre laser. La précision de la mesure sur les théodolites munis de distancemètres est de l'ordre du dixième de millimètre à 100m.

1.3.1.2 Laser à temps de vol

Les laser à temps de vol se contentent de mesurer le temps que met le rayon laser pour faire un aller/retour entre l'instrument et le réflecteur. Ils sont donc nettement plus rapides. Compte tenu de la précision voulue, la mesure du temps de vol ne peut être faite avec une horloge à quartz de faible précision. Les scanners laser utilisent des condensateurs temps/tension pour mesurer le temps de vol. Un condensateur déchargé se charge pendant le trajet du laser et on mesure la tension à ses bornes au moment du retour de l'impulsion. La charge du condensateur au cours du temps est :

$$U_c(t) = U_0 \cdot (1 - e^{-\frac{t}{\tau}})$$

où U_0 est la tension appliquée aux bornes du circuit et τ est la constante de temps du circuit Les imperfections du condensateur font que la conversion de la tension en temps de vol produit un bruit gaussien assez important. Pour limiter ce bruit, certains appareils permettent de répéter la mesure de distance plusieurs fois de suite avec la même orientation du rayon afin de moyenner les mesures. La vitesse d'acquisition est alors ralentie d'autant.

En une seule mesure la précision est d'environ ± 1 cm. La portée maximale est de l'ordre du kilomètre avec les lasers habituels. Pour augmenter la portée il faut augmenter la puissance du laser, ce qui rend nécessaire des mesures de sécurité contraignantes (interdiction de présence sur la zone balayée compte tenu du danger encouru sur la rétine de l'œil). La vitesse d'acquisition est de l'ordre de dix mille points par seconde. La grande vitesse et la portée de ce type d'instrument en fait le seul utilisable pour des acquisition de falaises instables. C'est donc ce type d'instrument qui a été utilisé dans le cadre de cette thèse et auquel il est fait référence dans la suite de ce mémoire.

À cause de la divergence du rayon laser, il peut arriver qu'une partie du rayon soit

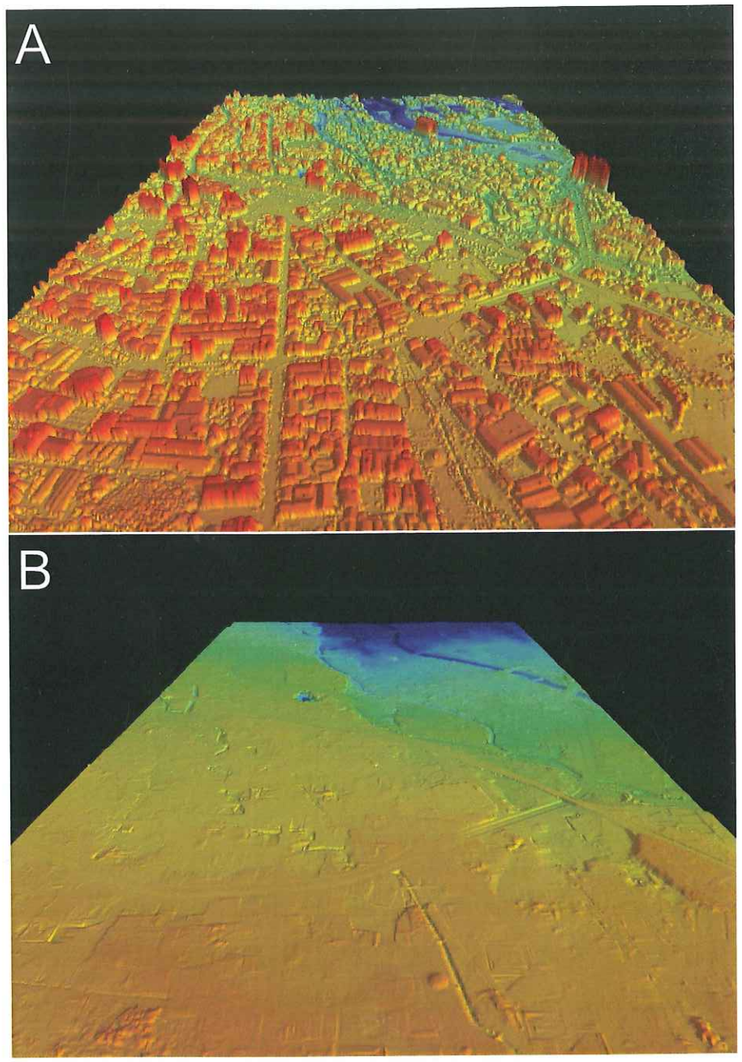


Fig. 1.14 – Différence entre modèle numérique de surface (MNS) et modèle numérique de terrain (MNT).
Sur le MNS (A), la végétation et les bâtiments sont présents, l'altitude du sol sous la végétation n'est pas connue. Sur le MNT (B), la végétation et les bâtiments sont absents, L'altitude du sol est indiquée partout, par interpolation là où elle n'est pas directement mesurable (le sol est supposé à la même altitude dans le bâtiment et à l'extérieur). D'après Chen (2007)

réfléchie par un objet au premier plan tandis que le reste du rayon ne soit réfléchi que par un objet plus lointain. Les premiers scanners ne prenaient comme temps d'arrivée que le premier retour, ignorant les objets à l'arrière plan. Le nuage de points obtenu correspondait alors à un modèle numérique de surface. Les scanners actuels permettent en général de choisir entre le premier et le dernier retour. Lorsqu'on enregistre le dernier retour, et sous réserve que la végétation soit suffisamment peu dense, le nuage de points obtenu correspond à un modèle numérique de terrain (MNT), dans lequel la végétation n'apparaît déjà plus (fig. 1.14). Il existe également des scanners qui enregistrent l'intégralité du signal retour. Cette information permet, lorsque le rayon laser parvient jusqu'au sol, de connaître en même temps l'altitude du sommet des arbres, de la végétation basse et du sol.

1.3.1.3 Mesure par triangulation

Un troisième type de scanner laser existe, ayant une précision et une vitesse intermédiaire entre la mesure par temps de vol et la mesure de phase. Ces scanners fonctionnent par triangulation. L'émetteur émet toujours un rayon laser, mais le récepteur est maintenant déporté à une distance connue. Il s'agit d'une caméra qui détecte le point rouge visible à l'endroit où le rayon laser rencontre la surface à étudier et mesure la direction d'où il provient. Connaissant la longueur L de la base, et les angles θ et α du dispositif (fig. 1.15), on peut calculer la position de la surface par

$$R = \frac{L}{\sin\theta + \cos\theta \cdot \cot\alpha}$$

$$X = R\sin\theta \cdot \cos\phi$$

$$Y = R\cos\theta \cdot \cos\phi$$

$$Z = R\sin\phi$$

Au fur et à mesure que l'angle θ varie, l'angle α selon lequel la caméra perçoit le point rouge du laser varie également. Ce scanner ne fonctionne que si le point rouge

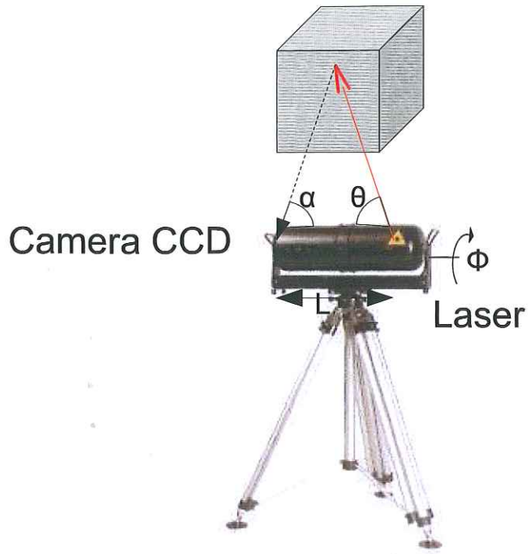


Fig. 1.15 – Mesure par triangulation lidar. La détermination de la distance est faite par un calcul trigonométrique à partir des angles θ, α et ϕ.

du laser est bien identifiable, ce qui nécessite en général l'absence d'autres sources de lumière. Les acquisitions sur sites naturels doivent donc être réalisées de nuit. La portée maximale du dispositif est de 25m et sa vitesse d'acquisition de 100 points par secondes, rendant impossible l'acquisition de la totalité d'une falaise instable avec ce type d'instrument [Charles, 2001].

1.3.2 Chaîne de traitement des données de scanner laser

Pour passer des données brutes issues du scanner laser, à des données simplement utilisables pour l'étude d'un site, plusieurs opérations sont nécessaires. L'ordre dans lequel on réalise ces différentes opérations a une influence sur le temps de traitement

nécessaire et la qualité du résultat final.

1.3.2.1 Géoréférencement et fusion des nuages de points

Il s'agit d'être capable de situer l'acquisition laser par rapport à un référentiel connu, ou pour le moins de positionner différentes acquisitions les unes par rapport aux autres si plusieurs points de vue ont été nécessaires pour couvrir l'ensemble de l'objet étudié. Plusieurs méthodes de référencement sont utilisables.

1. Un référencement à priori. Cette méthode est principalement utilisée pour les systèmes lidar embarqués, quand la position du lidar change au cour de l'acquisition. La position et l'orientation du lidar est déterminée à tout instant par un système de positionnement GPS couplé à une centrale inertielle. Cette méthode complexe à mettre en œuvre sera partiellement détaillée en 1.3.3
2. Un référencement par cibles. Des cibles seront positionnées dans le champ d'acquisition du scanner. Ces cibles doivent être facilement identifiables dans le nuage de points provenant de la scannerisation lidar et leur position précise doit pouvoir être déterminée précisément à l'aide d'une autre méthode, comme une visée au théodolite. Deux types de cibles sont couramment utilisées, celles identifiables par leur réflectivité et celles identifiables par leur forme.

   Les cibles identifiables par leur réflectivité sont soit des prismes soit des cibles planes en matériau rétroréfléchissant. Il est aisé de mesurer le centre de telles cibles avec un théodolite. Leur identification dans le nuage de points est faite en fonction de l'intensité du signal retour, à saturation pour ces matériaux rétroréfléchissants, alors qu'elle reste à des valeurs modérées pour les surfaces usuellement rencontrées. Lorsque la scène que l'on souhaite scanner est particulièrement réfléchissante, une solution symétrique peut-être adoptée en plaçant des cibles noires dont la réflectivité très faible sera identifiable par contraste avec les autres don-

nées.

Les cibles identifiables par leur forme sont généralement des sphères, dont le centre peut être déterminé à partir de la zone du nuage de point correspondant à la sphère. Elles ont l'avantage d'être facilement identifiables et d'être identiques quel que soit le point de vue. Par contre leur centre est difficilement mesurable avec un théodolite. Pour réaliser cette opération, il faut placer ces sphères sur une embase de trépied et qu'elles soient ensuite remplacées par une cible visée par théodolite ou par une antenne GPS.

Lorsque le référencement est réalisé avec des cibles, il est important d'identifier la position précise des cibles avant toute opération sur le nuage de points (réduction du bruit, triangulation, diminution du nombre de points etc). En effet, ces opérations peuvent modifier la position des points ou éliminer des points de la cible, et la précision de la position de la cible en est diminuée. Seule l'élimination de points aberrants peut être réalisée au préalable.

3. Le référencement par reconnaissance de forme. Cette méthode ne permet pas un géoréférencement complet, mais seulement un référencement de différentes acquisitions d'un même objet les unes par rapport aux autres. Le principe est d'utiliser une partie de l'objet présente sur deux acquisitions, et de trouver les paramètres de rotation et de translation d'une acquisition par rapport à l'autre qui minimisent l'écart entre les deux nuages de points pour cet partie de l'objet. Le calcul se fait par itérations, soit en minimisant l'écart entre chaque point du premier nuage et le point qui en est le plus proche dans le second nuage [Besl et McKay, 1992], soit en minimisant l'écart entre les surfaces qui sont interpolées à partir de ces deux nuages de points [Chen et Medioni, 1992]. Il est cependant nécessaire d'identifier préalablement des points communs aux deux acquisitions afin que l'algorithme convergence rapidement.

Le référencement par cible est le plus précis lorsque des cibles peuvent être placées loin les unes des autres et non alignées. Le référencement par reconnaissance de forme demande une grande puissance de calcul et possède l'inconvénient de ne pas fournir d'estimateurs de sa précision, mais il est parfois le seul utilisable, quand la paroi étudiée n'est pas suffisamment accessible et qu'aucune cible ne peut être placée dans le champ du scanner.

1.3.2.2 Élimination des points aberrants

Il arrive fréquemment que des points aberrants soient présents dans l'enregistrement lidar, souvent en avant de la paroi étudiée à cause de la réflexion par un élément étranger à la paroi (poussière, insecte, micro-goutte d'eau, etc.). Il peut arriver qu'il y ait des points en arrière de la paroi à cause d'un phénomène de réflexion multiple (fig. 1.16). Ce dernier phénomène est cependant rare et une partie des algorithmes d'élimination des points aberrants le néglige. Pour déterminer les points aberrants, différents algorithmes existent, souvent développés au départ pour les scannerisations aériennes. Une comparaison des principaux algorithmes est faite par Sithole et Vosselman (2003). Ces auteurs différencient quatre principaux concepts qui sous-tendent les algorithmes de filtrage :

1. les algorithmes basés sur la pente : si l'angle entre deux points successifs dépasse un seuil fixé, le point le plus proche est supposé être au dessus du sol dans le cas d'une scannerisation aérienne verticale, en avant de la paroi dans notre cas de figure.
2. les algorithmes à block minimum : si le point est trop loin du plan moyen formé par les points proches de lui, il est éliminé.
3. les algorithmes basés sur une surface : si le point est trop loin de la surface extrapolée à partir des points les plus proches, il est éliminé.

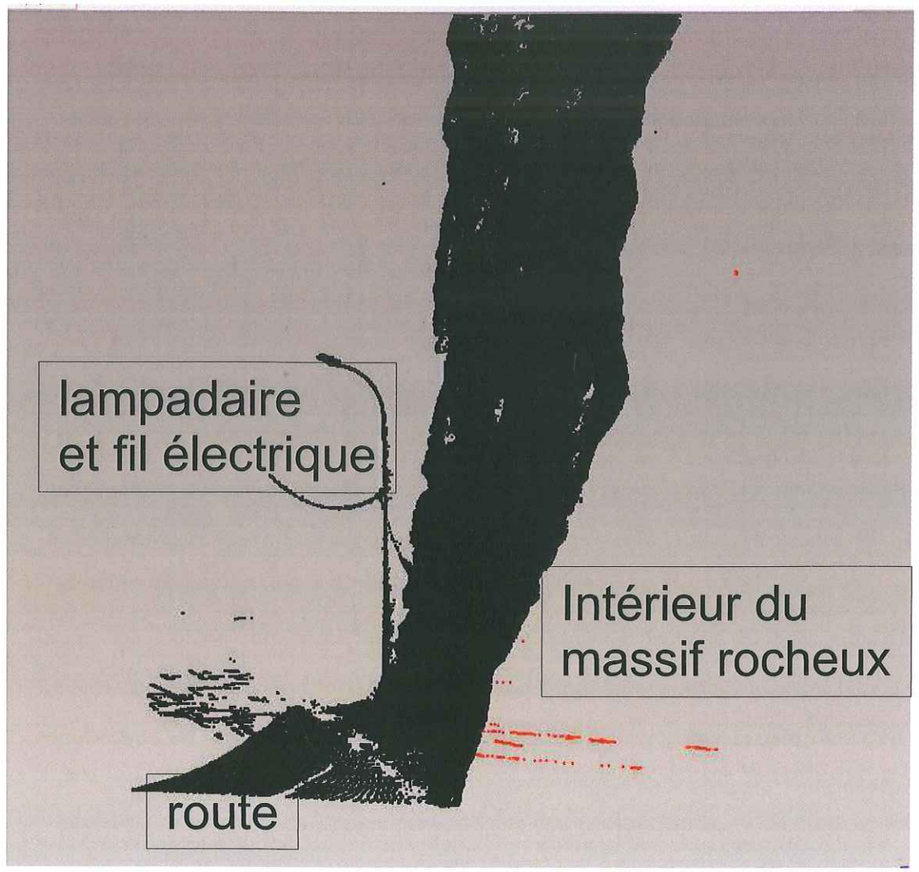


Fig. 1.16 – Présence de points aberrants en arrière de la paroi lors d'une scannerisation laser (moyenne corniche entre Nice et Eze).
Ces points s'expliquent par des trajets multiples entre la paroi et de bons réflecteurs situés en avant de celle-ci. Sur le cas présenté, il s'agit des panneaux de signalisation routière.

4. les algorithmes par cluster : les points sont regroupés en cluster par des algorithmes de classification statistique des individus. Si un cluster est devant un autre, c'est qu'il provient d'un objet au dessus du sol, ou en avant de la paroi et il est éliminé.

Même si la qualité relative des différents algorithmes dépend de la configuration rencontrée, les algorithmes basés sur une surface sont souvent les plus efficaces. Ce sont aussi les algorithmes les plus gourmands en ressources. La classification des points en points corrects ou aberrants peut être réalisée en une seule étape afin d'avoir un algorithme plus rapide, ou par itération (les points considérés comme aberrants ne sont plus pris en compte pour la détermination du statut des autres points) pour assurer une meilleure qualité du filtrage.

1.3.2.3 Réduction du bruit

Comme nous l'avons vu en 1.3.1, les laser à temps de vol sont soumis à un fort bruit gaussien sur la mesure de la distance. Ce bruit peut être réduit par des algorithmes de type filtre médian. Si certains algorithmes sont fait en prenant comme paramètres les caractéristiques techniques du lidar (écart type sur la distance, précision angulaire, déviation du faisceau), la plupart des algorithmes implémentés dans les logiciels commerciaux présupposent le niveau de bruit à partir d'une estimation de la dispersion des données. Dans le cas d'une surface rugueuse, la rugosité sera généralement perçue comme du bruit et sera éliminée ou réduite par les algorithmes de filtrage du bruit.

Les algorithmes diffèrent aussi sur la conservation du nombre de points entre données en entrée et données en sortie. Certains algorithmes déplacent les points en direction du plan moyen local sans supprimer de points, d'autres lissent le bruit en faisant la moyenne de n points et ne conservent que le point moyenné, diminuant ainsi la ré-

solution du nuage de points. La réduction du bruit peut souvent être réalisée en même temps que l'élimination des points aberrants. L'algorithme estime alors en fonction de la distance entre le point et son voisinage si il est trop loin, auquel cas il est éliminé, ou suffisamment près, auquel cas il est rapproché de la position « logique » qu'il devrait occuper (moyenne ou extrapolation de la surface suivant le type de filtre utilisé). Dans les scènes constituées de plusieurs nuages, il existe en plus du bruit lié à la mesure du laser, une rugosité apparente liée à l'erreur résiduelle lors du référencement des nuages de points les uns par rapport aux autres. Cette rugosité peut être traitée comme un bruit. Les algorithmes basés sur une surface sont aussi les mieux à même de diminuer ce type de bruit qui ne s'exprime pas selon une direction privilégiée.

Une autre approche a été adoptée récemment par Rabbani *et al.* (2007) pour limiter ce bruit : au lieu de procéder d'abord à une étape de référencement des acquisitions les unes par rapport aux autres, puis une étape de diminution du bruit, et enfin une étape d'extrapolation de la surface par triangulation, on procède en une seule étape. Dans la zone commune aux deux acquisitions, le référencement se fait en minimisant la distance entre les points des deux nuages et la surface interpolée à partir des deux nuages de points.

1.3.2.4 Triangulation

L'interpolation de la surface à partir du nuage de points peut être faite de différentes manières. Si on dispose d'un maillage régulier on peut opter pour une quadrangulation, mais en général la surface sera interpolée par triangulation. Les différentes techniques reposent sur la triangulation de Delaunay et son dual, le diagramme de voronoï.

Si S est un ensemble de points, la cellule de Voronoï d'un point $P \in S$ est l'ensemble

des points de l'espace plus proches de P que de tout autre point de S

$$V_{or}(P) = \{x \in E / \forall Q \in S; d(x,P) \leq d(x,Q)\}$$

La triangulation de Delaunay est définie à partir du diagramme de Voronoï de la manière suivante : si deux cellules de Voronoï $V_{or}(P)$ et $V_{or}(Q)$ ont une frontière en commun alors une arrête relie les deux points P et Q. En deux dimensions L'ensemble des arêtes ainsi définies forment une triangulation de Delaunay de l'ensemble S des points. En trois dimensions cette triangulation aboutit à un maillage par des tétraèdres du volume contenu dans le nuage de points. Dans le cas d'un scan laser, le nuage de points est une discrétisation de la surface, il faut donc ne conserver qu'une partie des arêtes ainsi définies pour obtenir une triangulation de la surface et non du volume. Il faut donc définir des conditions supplémentaires pour déterminer les arêtes à conserver et celles à rejeter. Une revue des différents algorithmes de triangulation à été faite par Bern (2004). Lorsque le nuage de point n'est pas suffisemment dense par rapport au rayon de courbure de la surface réelle, il peut y avoir des différences importantes entre la surface réelle et la surface modélisée ainsi qu'entre les surfaces issus de différents algorithmes. La différence peut porter sur la position de la surface à un endroit, qui varie en fonction des points à partir desquels elle est interpolée, et dont un exemple parlant est donné sur la figure 3.3. Cette différence peut aussi porter sur la présence ou l'absence d'un bord permettant de relier les faces internes et externes de l'objet. La possibilité de laisser des trous dans la surface, qui est justifiée par la possibilité d'avoir un accès à la face arrière de l'objet (par exemple une porte ouverte sur l'intérieur du bâtiment) ou de traverser l'objet (une fissure ouverte qui débouche sur le sommet de la falaise) aboutit à créer de nombreux trous dont la plupart n'ont pas de réalité physique dans les zones sous-échantillonées.

Les algorithmes de triangulation sont de plus conçus pour traiter tous le nuage de points d'un bloc, le diagramme de Voronoï devant être calculé de manière globale.

Il est de ce fait problématique de trianguler les dizaines de millions de points issus d'une scannerisation laser, et la plupart des algorithmes rendent nécessaire la réduction du nombre de point avant la triangulation, par exemple lors de l'étape de réduction du bruit. Pour résoudre ce problème Dey *et al.* (2001) proposent un algorithme qui subdivise le nuage de points en secteurs. Le diagramme de Voronoï sera calculé pour chaque secteur, et non globalement pour l'ensemble du nuage. La difficulté principale, résolue dans cet algorithme, est de trianguler de manière cohérente les frontières entre les différents secteurs.

Des algorithmes comme celui proposé par Dey et Goswami (2006) permettent de calculer une surface à partir d'un nuage de points bruité. Au lieu de réduire le bruit du nuage de points, puis de faire une triangulation s'appuyant sur les points du nuage débruité, la surface est interpolée à partir du nuage bruité sans que les points du nuage soient des sommets des triangles de la surface. Cette approche permet de réduire le temps de calcul global tout en ayant une réduction du bruit cohérente non seulement localement mais aussi globalement.

1.3.3 Scanner lidar héliporté

Il est souvent difficile de trouver des points de vue terrestres permettant de scanner une paroi rocheuse efficacement sur toute la zone intéressante. L'utilisation d'un scanner embarqué sur un hélicoptère, réalisant une acquisition depuis une position frontale par rapport à la paroi, est une alternative permettant de résoudre ce problème. Dans les systèmes héliportés ou aéroportés, le scanner laser est un scanner 2D ne balayant que le plan perpendiculaire à la direction de vol. La couverture de l'ensemble de la zone d'étude est réalisée parce que le porteur, avion ou hélicoptère, se déplace pendant l'acquisition, avançant suivant la direction X pendant que le scanner réalise une ligne

d'acquisition dans le plan YZ (fig. 1.17). Sur le site du Rocher de la Bourgeoise (cf. 2.3), le scanner était placé sur le coté de l'hélicoptère, dirigé à l'horizontale, et balayait des lignes de points entre 40° au dessus de l'horizon et 40° en dessous. La vitesse de l'hélicoptère est choisie, en fonction de la vitesse d'acquisition d'une ligne, du nombre de points par ligne et de la distance entre l'hélicoptère et la paroi, pour que l'écart entre deux lignes de points soit en moyenne égal à l'écart entre deux points de la même ligne.

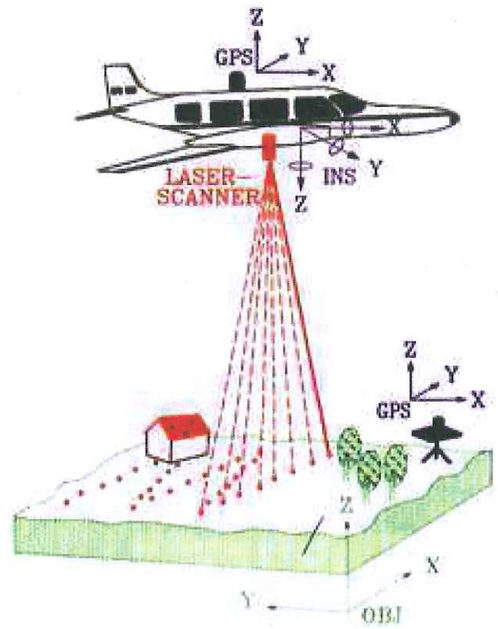


Fig. 1.17 – Fonctionnement d'un scanner lidar aéroporté.
Le lidar balaie des lignes perpendiculaires à l'avion. Le déplacement de l'avion fait que le spot laser échantillonne toute la surface. Un GPS et une centrale inertielle (IMU) doivent compléter le dispositif pour connaitre la position des points laser en absolu et non seulement en relatif par rapport à l'avion. D'après Chen (2007)

Le lidar héliporté possède plusieurs spécificités par rapport au lidar terrestre. La plus évidente d'entre elle est que l'instrument n'est pas fixe pendant l'acquisition. Il

est de ce fait nécessaire de déterminer la position et l'orientation de l'instrument à une fréquence identique à la fréquence d'acquisition des points lasers (quelques kHz). Cette détermination est réalisée par le biais d'un GPS différentiel, qui permet de connaitre précisément la position à intervalle régulier (entre 1 et 5 Hz), et d'une centrale inertielle, qui permet d'une part d'interpoler la position entre deux pas de temps de l'enregistrement GPS et d'autre part de calculer l'orientation du système. Le traitement du GPS différentiel n'étant pas réalisé en temps réel, le système n'enregistre pas la position des points lidar au fur et à mesure, mais stocke les données brutes du lidar (distance, orientation interne et réflectivité), du GPS (code et phase du signal) et de la centrale inertielle (accélérations et rotations subies). L'ensemble des données est synchronisé sur le temps GPS de la mesure, la précision temporelle étant de quelques μs. Des systèmes avec 3 systèmes GPS permettent de recaler la dérive de la centrale inertielle non seulement sur sa position mais aussi sur l'attitude du système. La précision de tels systèmes est fonction de la distance entre les antennes et de la rigidité de l'ensemble.

La précision du système est liée à plusieurs sources d'erreur :

- L'erreur gaussienne sur la mesure de distance par le laser, comme précédemment.
- L'erreur sur la détermination de la position par le GPS, elle dépend essentiellement de la qualité de la constellation GPS observée. Elle est de l'ordre de 2 à 20cm, l'incertitude sur la composante verticale étant supérieure à celle sur la position horizontale. Cette erreur se répercute directement sur la position des points, et n'est donc pas fonction de la distance entre l'hélicoptère et la paroi.
- L'erreur sur l'attitude du système et la détermination précise de la position entre deux positions GPS, due à la dérive de la centrale inertielle. Elle est proportionnelle au pas d'acquisition GPS pour les paramètres pouvant être recalés et au temps de vol pour les paramètres n'étant pas recalés. Elle varie beaucoup en fonction de la qualité de la centrale. Pour un composant de moyenne gamme, cette dérive est de quelques centimètres et de quelques dix-millièmes de degrés par seconde.
- L'erreur sur l'orientation du miroir du laser, augmentée d'une erreur dynamique sur la planéité du miroir. En effet sous l'effet des vibrations, le miroir du laser se déforme partiellement. Cette erreur est très difficile à estimer.

À cause de la méthode d'acquisition par déplacement du porteur, certains des algorithmes présentés en 1.3.2.2 et1.3.2.3 ne s'appliquent plus, ou moins bien. Le faisceau laser balaye des lignes verticales avec un pas angulaire constant, la variation d'assiette et de position de l'hélicoptère entre deux émissions du rayon laser étant négligeable. Par contre l'écart spatial entre deux lignes est variable à cause des changements de direction et de vitesse de déplacement de l'hélicoptère. À cause de ce pas métrique et variable en X, les méthodes de filtrages reposant sur une comparaison avec les points proches ne fonctionne plus sauf à avoir une maille de filtre adaptative.

1.4 Limites des méthodes

La photogrammétrie argentique ou numérique, le lidar terrestre ou héliporté ont donc des spécificités propres qui font que leurs limites en termes de résolution, précision, temps de travail sont très différents. Ces différences sont détaillées dans les paragraphes qui suivent.

1.4.1 Résolution

Les limites de résolution des différentes méthodes sont des limites de résolution angulaires. On peut toujours obtenir une meilleure résolution en se rapprochant de

l'objet étudié. Malheureusement il n'est pas toujours possible de se rapprocher à cause de la dangerosité du site ou de l'absence de point de vue. Il peut aussi ne pas être souhaitable de le faire afin de garder une acquisition en un seul point de vue et d'éviter les problèmes de coréférencement de différentes acquisitions. La résolution angulaire de la photogrammétrie est fonction de la focale de la lentille utilisée et de la dimension du pixel. Pour avoir un large base en conservant un bon taux de recouvrement entre les deux photographies, il faut que la photographie couvre un angle total de l'ordre de 60 degrés. Avec cette ouverture la résolution angulaire est d'environ 20 millièmes de degrés pour un appareil numérique haut de gamme ou un chambre photogrammétrique moyen format, et d'environ 2 millièmes de degrés pour une chambre photogrammétrique grand format.

Lorsque l'on compte utiliser uniquement la photogrammétrie avec corrélation automatique, la distance entre les deux prises de vue peut être plus faible, on peut donc utiliser une focale plus grande afin d'avoir une meilleure résolution angulaire. On peut multiplier par 1,5 la résolution angulaire sans trop perdre en couverture de l'objet parce que le taux de recouvrement entre les photographies est plus élevé. Au delà le champ couvert devient trop faible et l'augmentation de la focale ne présente plus d'intérêt.

Les scanners lidar terrestres ont des résolutions angulaires maximales de 2 à 4 millièmes de degrés suivant les appareils. Ils sont souvent utilisés à des résolutions plus faible pour des raisons de durée d'acquisition. La résolution effective de scannerisation est par ailleurs souvent réduite lors de la réduction du bruit. Les lidar héliportés ou aéroportés ont une résolution angulaire de quelques centièmes de degrés entre deux points successifs.

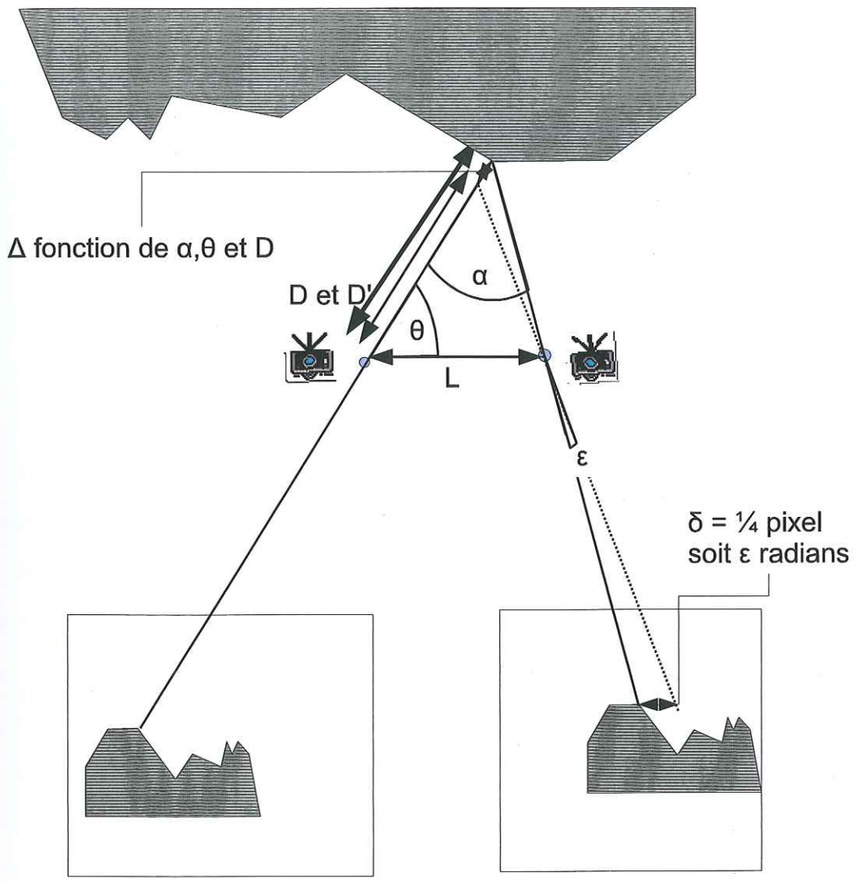


Fig. 1.18 – Précision de la mesure par photogrammétrie.
Une incertitude d'un quart de pixel correspond à une incertitude absolue fonction de l'écart angulaire α entre les rayons, et de l'angle θ entre le rayon de gauche et la base. plus α est petit, plus l'incertitude est grande.

1.4.2 Précision sur la position d'un point

Il ne sera pas abordé ici le problème de la précision du calage du modèle, photogrammétrique ou issu d'une scannerisation laser, par rapport à des coordonnées terrains connues. Cette précision est en pratique plus limitée par la précision de la mesure terrain des points d'appuis et leur trop petit nombre que part la précision intrinsèque de l'instrument de mesure. D'autre part, l'erreur de calage n'a pas d'influence sur les mesures de position relative entre deux points donnés du modèle.

La précision du pointé photogrammétrique est de 0.29 pixels en manuel [Thom, 2002]. En corrélation automatique la précision en termes de pixel s'améliore lorsque les photographies sont prises avec une base courte. Dans ces conditions une même erreur angulaire correspond à une incertitude plus grande en terme de distance, car la valeur de α diminue dans l'équation (1.15). La précision en terme de distance est donc à peu près la même. Cette précision peut égaler la précision d'un pointé visuel, mais est soumise aux artéfacts de corrélation, qui ne sont pas forcément détectables.

Une erreur d'un quart de pixel correspond à une erreur angulaire de 0,005° pour les appareils numériques et 0,0005° pour les chambres optiques grand format, dans les conditions de résolution maximale évoquées précédemment. Comme le montre la figure 1.18, la précision métrique correspondante est fonction :

1. de l'angle entre le rayon issu du point de vue de droite et celui issu du point de vue de gauche

2. de la distance entre le point de vue et l'objet

En considérant que la position du détail est fixée dans la photographie de gauche et

que la recherche du pixel homologue se fait sur la photographie de droite, on a :

$$D = L\frac{\sin(\theta+\alpha)}{\sin\alpha}$$
$$\text{et } D' = L\frac{\sin(\theta+\alpha\pm\epsilon)}{\sin(\alpha\pm\epsilon)}$$

en développant au premier ordre en ϵ, on obtient, avec ϵ en radians :

$$D' = L\frac{\sin(\theta+\alpha)}{\sin\alpha}\left(1\pm\frac{\epsilon}{\tan(\theta+\alpha)}\mp\frac{\epsilon}{\tan\alpha}\right)$$
$$\text{soit } \Delta = L\frac{\sin(\theta+\alpha)}{\sin\alpha}\left(\frac{\epsilon}{\tan(\theta+\alpha)}-\frac{\epsilon}{\tan\alpha}\right)$$
$$\Delta = D\left(\frac{\epsilon}{\tan(\theta+\alpha)}-\frac{\epsilon}{\tan\alpha}\right) \quad (1.15)$$

Dans la pratique, certains logiciels travaillent plutôt en déplaçant les coordonnées terrain du point recherché parallèlement à la base photogramétrique, ce qui déplace simultanément la zone de corrélation sur la photographie de droite et sur la photographie de gauche. L'incertitude d'un quart de pixel est alors répartie entre les deux photographies. L'erreur sur la position terrain avec cette deuxième méthode est un peu plus faible dans la zone centrale et un peu plus grande sur les bords des photos, le terme en $\tan(\theta+\alpha)$ de l'équation 1.15 est légèrement modifié, mais le terme en $\tan\alpha$ est conservé, la différence étant de second ordre par rapport à la valeur de l'erreur. Au milieu de la zone de recouvrement entre les deux photographies, pour une base égale à la distance entre les points de prises de vues et l'objet, on a $\alpha \approx 54°$ et $\theta \approx 63°$. ce qui donne une erreur de 0,009% de la distance pour une appareil numérique et de 0,000 9% pour une chambre photogrammétrique grand format

Pour les scanners lidar terrestres, la précision est liée à la précision de la mesure de l'angle d'émission du rayon laser, à la divergence du rayon laser et au bruit gaussien sur la mesure du temps de trajet, l'écart entre la vitesse de propagation de la lumière dans

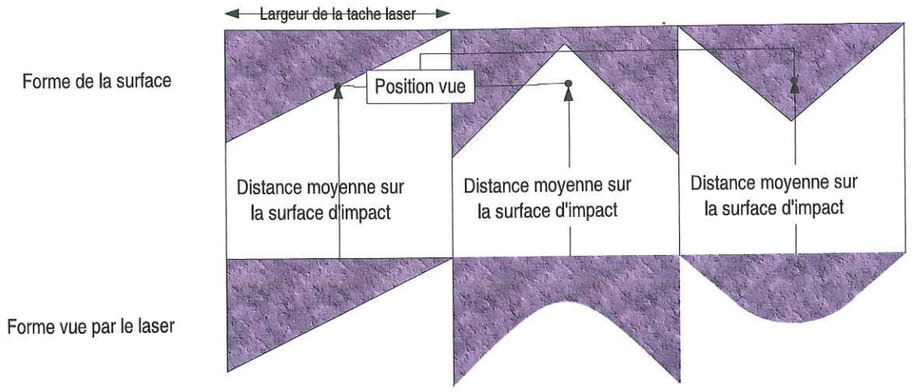


FIG. 1.19 – Effet de la forme de la surface sur la forme du pulse retour.
une paroi convexe est vue plus proche, une paroi concave est vue plus loin, une paroi plane est vue au bon endroit.

le vide et dans l'air étant négligeable. Pour les deux scanners que nous avons utilisés, La mesure d'angle est précise à 0,002° et la divergence du rayon laser est de 0,014°. Ce qui donne une erreur maximale sur la position latérale du point réflecteur de $2,8 \cdot 10^{-4}D$. L'effet principal de la divergence du rayon laser est de lisser les angles, son influence sur une surface plane est négligeable (fig. 1.19). Le bruit gaussien sur la mesure de distance possède un écart type $\sigma \approx 0,01$ à $0,02\%$. On a donc une précision sur la position du point d'environ 0,03% de la distance, soit 3cm à 100m. Cette précision est un ordre de grandeur au dessus de ce que l'on peut obtenir par photogrammétrie.

Pour les lidars héliportés ou aéroportés, les deux plus grandes sources d'erreurs sont l'erreur sur la détermination de la position par le GPS, qui varie entre 2 et 10 cm en planimétrie et entre 5 et 20 cm en altimétrie suivant la qualité de la constellation de satellite [Vallet, 2002], et l'erreur sur l'attitude du système dû à la dérive des centrales inertielles, qui est de quelques millièmes à un centièmes de degré pour un vol d'une dizaine de minutes. La précision finale dépend de la durée du vol de l'hélicoptère et de la distance entre l'hélicoptère et la paroi scannée. Typiquement, pour quelques minutes

d'acquisition et une distance de 100 m, la précision sera de 10 à 30 cm, soit encore un ordre de grandeur au dessus de ce qui peut être obtenu par laser terrestre.

1.4.3 Temps nécessaire à l'acquisition des données sur le terrain

Le temps d'acquisition est relativement court pour toutes ces méthodes. Pour la photogrammétrie le temps de prise de la photo est inférieur à la seconde, reste le temps d'approche et le temps de positionnement entre les stations. Le temps d'acquisition peut donc être considéré comme égal au temps d'approche. Il y a par contre un temps de préparation additionnel pour déterminer la position des points d'appuis. Cette préparation peut être faite avant les prises de vue, en traçant sur la paroi des marques qui seront visibles sur les photographies et en mesurant leurs positions. Elle peut être différée, après l'acquisition et le développement des clichés, en retournant sur le terrain mesurer la position de détails caractéristiques facilement identifiés sur les photographies. Que ce soit réalisé avant ou après la prise de vue, il faut compter une demi-journée à une journée de levés topographiques. Si un même site est étudié plusieurs fois de suite, cette préparation n'est nécessaire qu'une seule fois : les mêmes points d'appuis peuvent être réutilisés à condition de les avoir disposés sur des zones qui restent fixes. Une orientation directe par GPS et centrale inertielle peut être utilisée afin de supprimer ce temps de préparation au prix d'un surcoût en matériel et d'une perte de précision de l'orientation absolue.

Avec un lidar héliporté, l'opération est aussi relativement rapide. Le temps d'acquisition proprement dit dépend de la résolution mais reste relativement bref : pour $20000m^2$ de paroi (250 m de long par 80 m de haut) scanné à 80 m de distance à résolution maximale, il faut compter 5 minutes. À ce temps s'ajoutent approximativement

une heure pour le montage du système sur l'hélicoptère, l'installation d'une station GPS fixe à proximité immédiate du site étudié, et la même chose pour le démontage final. Aucune opération de géoréférencement supplémentaire n'est nécessaire, la position du lidar étant fournie par le GPS.

Avec un lidar terrestre, la vitesse d'acquisition en nombre de points par seconde est fixée. Elle dépend de l'instrument mais est de l'ordre de 2500 à 12 000 points par seconde. Le temps d'acquisition sera donc inversement proportionnel au carré de la résolution. Pour une scène de 40°×40° avec une résolution angulaire de 0,010°, ce qui correspond à une falaise de 80m par 80m scannée à 100m de distance avec un point tous les 3,5 cm, le temps d'acquisition sera de 30 minutes avec le laser Riegl LMS Z 420 et de 2 heures avec le laser Optech ILRIS. Avec la résolution maximale de 0,002° il faut 25 fois plus longtemps. De plus, comme pour la photogrammétrie il peut être nécessaire de mettre en place des cibles et de relever leur position si l'on veut utiliser un référentiel différent du référentiel propre au laser. Le temps nécessaire est d'une demi-journée de travail topographique, une journée si plusieurs points de vue doivent être raccordés sans l'utilisation de la reconnaissance de forme. Le travail topographique de levée des cibles peut en grande partie être réalisé pendant la durée de l'acquisition lidar. Le temps passé sur un site est en général d'une journée de travail complète, ce qui permet de réaliser deux à trois acquisitions avec une résolution de 0,005° (2cm à 100m), mais n'est pas suffisant pour une acquisition à pleine résolution.

Les acquisitions laser nécessitent donc nettement plus de temps sur le terrain qu'une scannerisation laser héliportée ou que la prise de vue photogrammétrique, mais restent d'une durée comparable à celle de la photogrammétrie y compris le temps de préparation.

1.4.4 Temps de traitement des données

Le temps de traitement des données variera beaucoup suivant que l'on choisit un traitement standard ou que l'on a besoin de développer des solutions spécifiques pour résoudre un problème particulier. Les temps donnés ci-dessous correspondent au temps nécessaire à l'obtention d'un modèle numérique de surface sur un site ayant des dimensions de l'ordre de 80 × 80m.

En photogrammétrie, que ce soit pour un traitement manuel ou automatique, il faut commencer par orienter le modèle photogrammétrique, soit quelques heures de travail par couple ne posant pas de problème particulier. Pour les photographies argentiques il faut ajouter le délai de développement et de numérisation des photographies, soit deux jours de délai. Il faut ensuite un à deux jours de travail pour obtenir un modèle numérique de surface de manière entièrement automatique, une semaine de manière semi-automatique. Pour des mesures directes sur le modèle photogrammétrique, il faut compter une petite demi-journée de travail pour mesurer l'orientation d'une quarantaine de plans de fracture.

Pour la scannerisation laser terrestre, il faut compter deux jours de calcul pour aboutir à un modèle numérique de surface, une demi-journée supplémentaire si différents points de vue sont à associer.

Pour la scannerisation laser héliporté, le temps de travail sur le nuage de point est le même que pour le laser terrestre, auquel il faut cependant ajouter le temps du traitement des données du GPS et de la centrale inertielle. Soit 3 à 4 jours au total pour obtenir le modèle numérique de surface.

Pour un nombre de mesures relativement réduit, à condition que l'acquisition soit numérique et que les points d'appuis aient été mesurés au préalable, la photogrammétrie

permet d'obtenir les résultats plus vite que les autres méthodes. Dans les autres cas les différentes méthodes présentent des temps de traitement similaires. Seul le temps d'obtention d'un modèle numérique de surface par photogrammétrie de manière semi-automatique est sensiblement plus long.

1.4.5 Difficultés particulières de réalisation des couvertures photographiques et laser

Le principal problème qui n'a pas encore été évoqué, aussi bien en photogrammétrie qu'avec les acquisitions de données lidar, est que la qualité de l'acquisition est sensible aux conditions météorologiques. En photogrammétrie les photographies seront d'autant plus faciles à exploiter, et la précision sera d'autant meilleure, que le contraste des détails de la paroi sera adaptée à la dynamique du capteur. Suivant l'orientation et la nature de la paroi, les conditions optimales varient. Ce peut être une journée particulièrement ensoleillée pour maximiser le contraste entre ombre et soleil mais c'est plus souvent une journée avec une couverture nuageuse d'épaisseur limitée et très diffusante. On est donc amené à trouver un compromis entre attente des conditions idéales, disponibilité du matériel et des opérateurs et urgence de l'étude. Le problème de l'éclairement naturel de la scène ne se pose pas avec le scanner lidar, mais des pluies ou du brouillard sont aussi des conditions défavorables voire rédibitoires. Alors qu'il est toujours possible de réaliser une couverture photographique pendant une courte fenêtre favorable, une acquisition lidar par temps changeant est bien plus problématique du fait de la durée des scannerisations laser.

Dans le même ordre d'idée, les acquisitions n'ayant pas un caractère urgent seront si possible programmées en période de faible couverture végétale (fin d'automne, hiver, début de printemps). Pour les parois en altitude et possédant des vires ou des zones peu

pentues, il faudra au contraire réaliser les acquisitions pendant l'absence de couverture neigeuse.

1.5 Conclusion

Deux questions se posent en parallèle, le choix du type de données à acquérir parmi les techniques présentées dans ce chapitre et le choix de la résolution et de la précision avec lesquels on souhaite utiliser ces techniques. Ce choix sera fortement dépendant du site à étudier, tant au niveau de l'applicabilité de l'une ou l'autre des méthodes qu'en ce qui concerne la précision et la résolution qui peuvent être matériellement atteintes. Ce choix dépendra aussi des contraintes de temps de traitement. La photogrammétrie reste à privilégier lorsqu'une très grande précision est nécessaire, ou que seule la position d'éléments caractéristique importe. La scannerisation laser est par contre plus rapide et plus complète quand on a besoin d'obtenir l'enveloppe de l'ensemble de la zone étudiée, par exemple pour servir de support à une simulation numérique. Nous allons voir dans les chapitres suivants la manière dont les contraintes imposées par les sites étudiés peuvent réduire la résolution théorique des méthodes que nous venons de présenter.

Chapitre 2

Sites étudies

Plusieurs sites ont été étudiés dans le cadre de cette thèse, de manière plus ou moins approfondie suivant les sites. Trois sites, le Rocher du Midi, le Ravin de l'Aiguille et le Rocher de la Bourgeoise, ont été étudiés dans le cadre du projet CAMUS (caractérisation multi-méthodes des aléas d'éboulement en masse), projet de recherche de deux ans financé dans le cadre programme Risque Décision Territoire (RDT) du Ministère de l'Écologie et du Développement Durable (MEDD). Sur ces sites une étude géophysique a été réalisée par une équipe du laboratoire de géophysique interne et tectonophysique en parallèle à notre étude de la topographie externe de la paroi. Ce projet avait pour objectif de mettre au point une méthodologie opérationnelle de reconnaissance permettant une description quantitative la plus précise possible des compartiments instables et une meilleure caractérisation de l'aléa d'écroulement en masse, point de départ de l'évaluation du risque. Un quatrième site, les Gorges de Paganin, a été étudié dans le cadre d'une action conjointe avec le bureau d'étude IMS. L'objectif final était dans ce cas de réaliser des simulations trajectographiques de chutes de blocs. Les trois sites du programme RDT sont situés près de Grenoble tandis que les gorges de Paganin se trouvent dans l'arrière pays niçois (fig. 2.1).

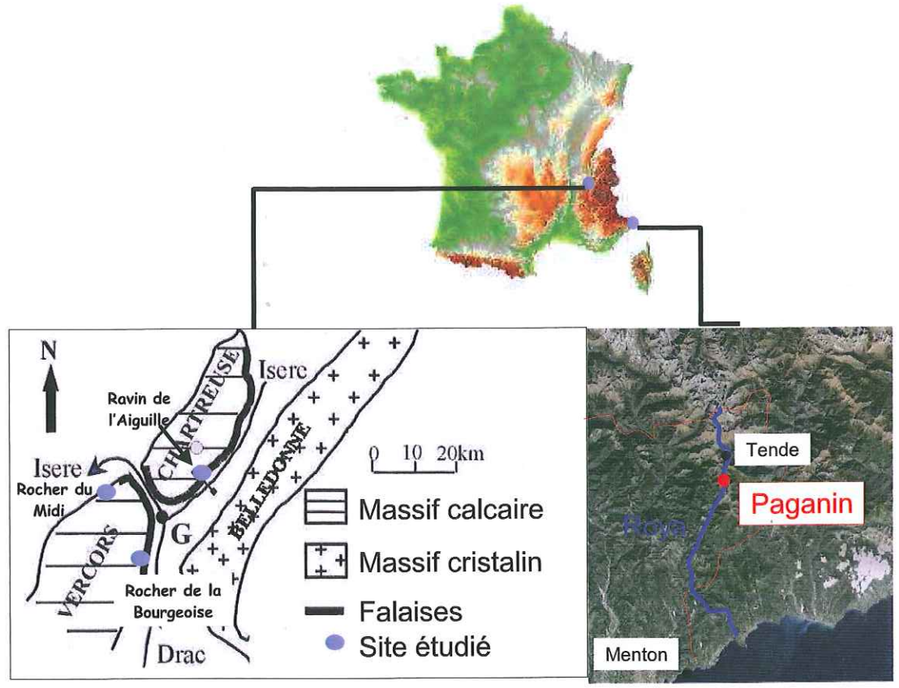


Fig. 2.1 – Plan de situation des sites étudiés.
Trois sites sont situés aux alentours de Grenoble : le Ravin de l'Aiguille dans le massif de la Chartreuse, le Rocher du Midi et le Rocher de la Bourgeoise dans le massif du Vercors. Le dernier site, les gorges de Paganin, est situé dans la vallée de la Roya

2.1 Le Rocher du Midi

2.1.1 Contexte géologique

Le premier site étudié a été le site du Rocher du Midi. C'est une falaise verticale de 150m de haut, orientée N-S qui domine le village de La Rivière sur le rebord ouest du Vercors (fig. 2.1 et 2.2).

Du point de vue géologique, la falaise est constituée de calcaire Urgonien inférieur, présentant un faciès calcaire à pâte fine ou cristalline, avec des variations latérales de faciès et des conditions de sédimentation peu profondes à péri récifales. La stratification présente un léger pendage Est, à contre pente. La faille chevauchante de Montaud, orientée NE-SW prend en écharpe l'ensemble du chaînon du rocher du midi et passe à 250m au SE du compartiment étudié. Aucune réplique de celle-ci n'est cependant visible sur le site (fig. 2.2). De nombreuses cicatrices d'éboulements sont visibles sur la falaise. On observe un éboulement ancien, situé immédiatement au nord du secteur étudié, d'un volume évalué entre $500000m^3$ et $1000000m^3$ ainsi qu'un éboulement récent de volume plus modeste, situé une centaine de mètres au sud de la zone d'étude (fig 2.3).

2.1.2 Données acquises sur le terrain

Le compartiment étudié fait 80m de haut sur environ 40m de large, il est limité par une importante fracture ouverte, quelques mètres en arrière de la paroi, d'orientation moyenne N130°E. La figure 2.3 montre les limites de la zone d'étude, avec, en bord gauche de la zone étudiée, la fracture qui se prolonge jusqu'à 65m de profondeur à partir du plateau jusqu'à une vire intermédiaire. La figure 2.6 montre bien l'étendue

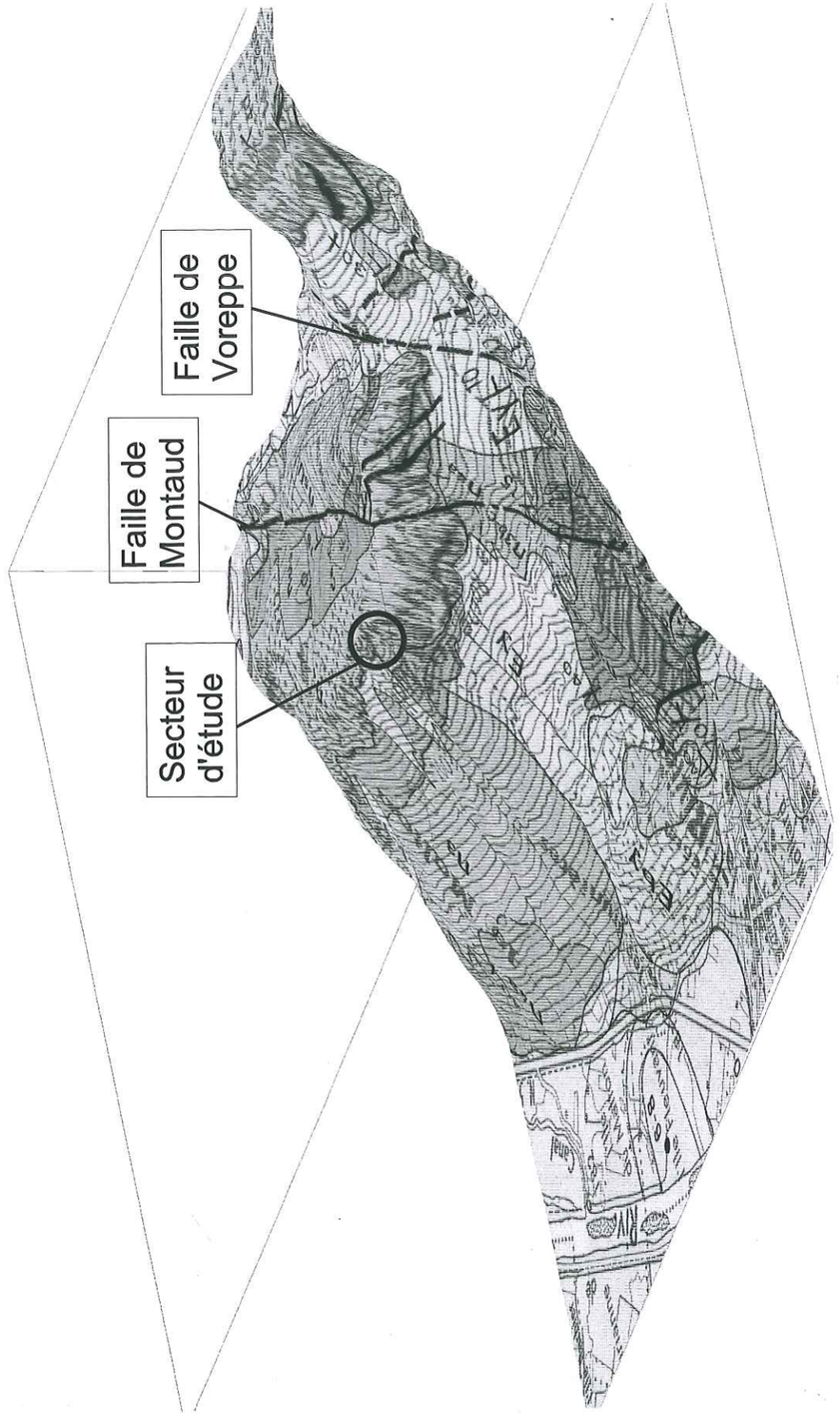


Fig. 2.2 – Contexte géologique du rocher du midi.
La zone étudié est entouré. La faille de Monteau, d'importance régionale, passe à quelques centaines de mètres.

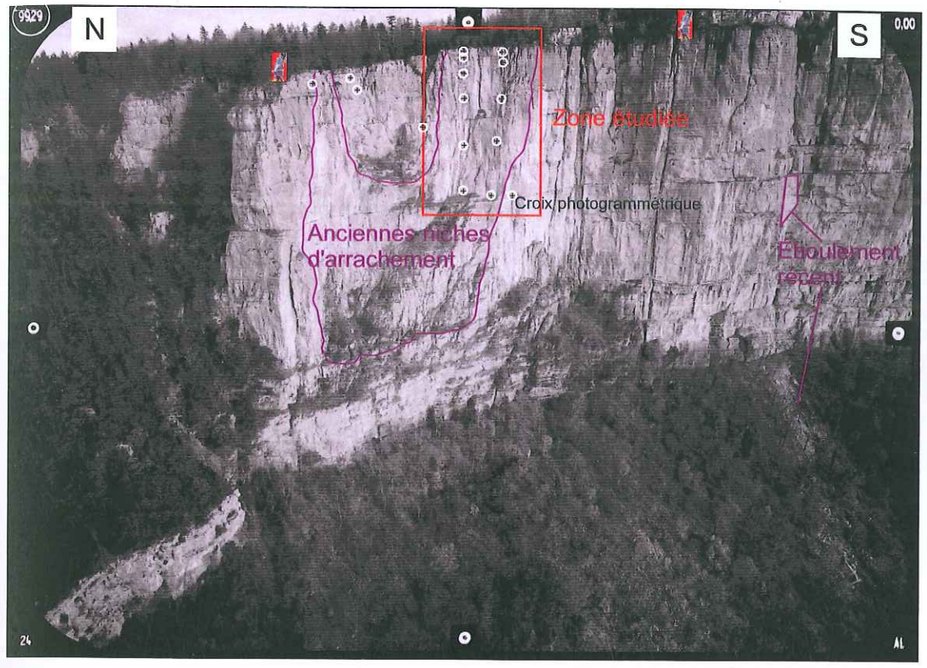


Fig. 2.3 – Photographie du site du rocher du midi
Photographie réalisé pour la couverture photogrammétrique du site. Le compartiment étudié est délimité en rouge. Les traits violets délimitent des anciennes niches d'arrachement. Les positions des points de vue laser, sur l'éperon nord et sur la vire sud, sont figurés.

de la fracture principale en arrière du compartiment étudié. Dans le détail, c'est une fracture en baïonnette formée par une alternance de plans de fracture subverticaux N110°E et N20°E.

Des mesures extensométriques réalisées depuis une dizaine d'années au sommet de cette fracture semblent montrer une tendance à l'ouverture avec une vitesse de l'ordre du mm par an (Didier Hantz, communication personnelle).

2.1.2.1 Relevé de fracturation

Un relevé des orientations des fractures visibles sur le site a été effectué à la boussole clinomètre. Un premier échantillon de 71 fractures a été mesuré sur le plateau au dessus de la paroi rocheuse. Un second échantillon de 24 fractures a été mesuré sur la paroi lors d'une descente en rappel [Jongmans *et al.*, 2007]. Pour ce second échantillon le pendage, subvertical pour l'ensemble des données, n'a pas pu être mesuré et l'horizontale est mesurée à 5° près par simple alignement de la boussole, en lieu et place d'un accolement direct sur la paroi. Les résultats, présentés sur la figure 2.4, montrent une famille N110°E à N150°E et une famille N20°E présentant quelques individus avec des orientations allant jusqu'à N55°E. Les joints de stratification sont rares et peu individualisés. Le seul exemple, mesuré sur le plateau, montre une orientation N44°E, 35°NW.

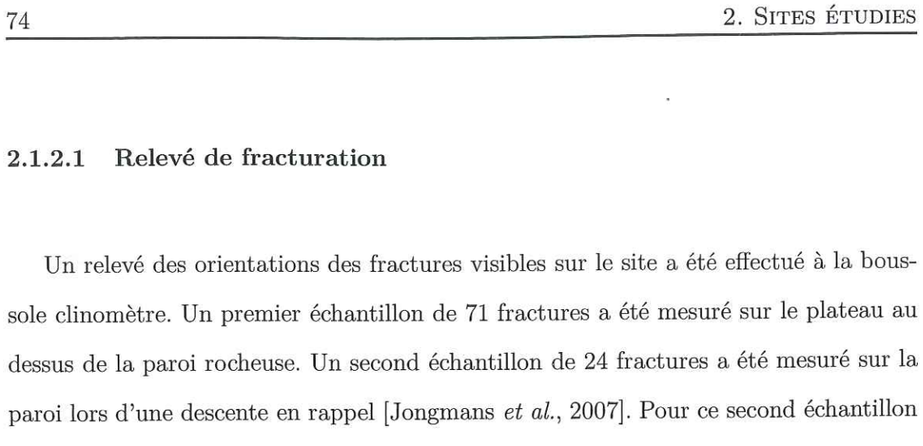


Fig. 2.4 – Relevé direct des familles de fracture du Rocher du Midi.
Les mesures ont été réalisés, de haut en bas, à la boussole en paroi et à la boussolesur le plateau. De gauche à droite sont présentées : les traces cyclographiques des plans, les pôles des plans, les courbes d'isodensité de pôles et les diagrammes en rose.

2.1.2.2 Couverture photographique

Une double série de prises de vue a été acquise sur le site : une série de quatre photos (24 à 27) avec un axe de prise de vue horizontal et une deuxième série de trois photos (47 à 49), depuis une altitude de prise de vue plus élevée et avec un axe oblique de manière à couvrir en même temps la paroi et son plateau sommital (fig. 2.5). La réalisation de ces prises de vue a été sous-traitée auprès du cabinet de géomètres experts Sintégra de Meylan. Il s'agit de clichés argentiques panchromatiques, au format

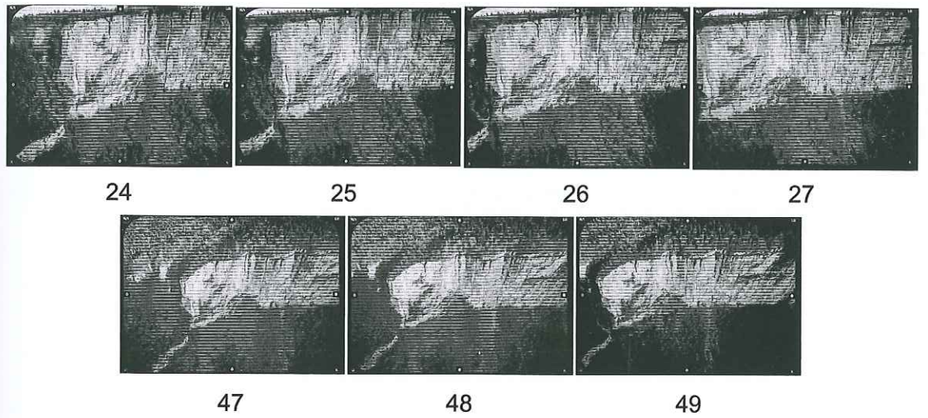


Fig. 2.5 – Couverture photographique du rocher du midi.
Une première série de photographie, 24 à 27, a été prise depuis l'altitude de la paroi. Une seconde série, 47 à 49, depuis une altitude plus élevée afin de couvrir le plateau.

18x13cm réalisés avec une focale de 100mm (Tab. 2.1). Les photographies ont été prises depuis un hélicoptère, à une distance moyenne de 200m de la paroi. L'échelle est donc le $1/2000^{ième}$ en moyenne. La couverture végétale s'est révélée trop importante pour que les vues du plateau soient exploitables. La figure 2.3 montre l'étendue de la zone d'étude sur une des photographies frontales. À 200m, une partie importante de la photographie est en dehors de la zone d'étude.

Les photographies ont été scannées à la résolution de $12\mu m$. La dimension de l'image

Paramètre	Valeur (mm)		Distance r (mm)	Distorsion δr (μm)
focale	99.29		15	4
repère n°1	(-80.388 ; 0)		25	7
repère n°2	(80.172 ; 0)		40	5
repère n°3	(0.001 ; 57.247)		50	2
repère n°4	(-0.003 ; -57.288)		70	-5

Tab. 2.1 – Données de calibration de la chambre photogramétrique UMK utilisé sur le site du Rocher du Midi

numérique est donc de 13000 × 9000pixels et la taille sur le rocher d'un pixel de 24mm en moyenne. Les couples ont été orientés à partir de repères tracés sur la paroi en tête de falaise et le long des profils géoradar (fig. 2.3). Ces repères sont des croix noires de 50cm à 1m d'envergure faites avec une bombe de peinture. Leurs positions précises ont été mesurées à l'aide d'une station totale.

2.1.2.3 Scannerisation lidar

L'acquisition lidar a été réalisée avec un lidar Riegl LMS Z420 (Tab. 2.2) depuis deux points de vue différents. Les deux points de vue, figurés sur le schéma 2.6, sont distants de 100m en moyenne de la zone étudiée. Ces points de vue permettent de couvrir 80% de la zone d'étude. Plus de 75% de la zone d'étude est couverte depuis le point de vue nord. Le point de vue sud couvrant uniquement 5 à 10% en limite sud du compartiment étudié. Le recouvrement entre les nuages de points issus des deux points de vue est très faible.

Les deux acquisitions ont été faites avec un pas angulaire de 20mgon (0,018°), ce qui correspond à une distance entre deux points de 1,5cm sur une surface perpendiculaire à la visée laser et de 4,4cm pour une surface présentant un angle de 20° avec le rayon laser. Pour un angle d'incidence de 1°, la distance entre deux points et de 1m environ. L'incidence du laser sur la paroi est très rasante, de l'ordre de 20° en moyenne. Elle

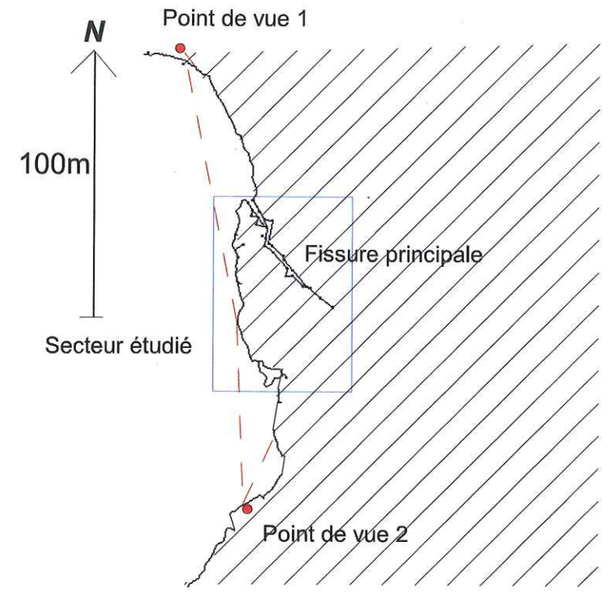


Fig. 2.6 – Plan de surface du rocher du midi.
L'intérieur du massif est hachuré. Les deux acquisitions laser ne se chevauchent presque pas. Le contour de la falaise est déduit des données laser.

Portée(Min, Max ($\rho = 10\%$), Max ($\rho = 80\%$)	2m, 350m, 1000m
Précision, écart type à 50m Répétabilité, écart type à 50m	10mm 8mm
Vitesse d'acquisition	12000 pts/s
Champ de visée	80°×360°
Divergence du laser	0.25mradians
Pas angulaire minimum	0.004°
Précision angulaire	0.002°

Tab. 2.2 – Données constructeur du laser Riegl LMS Z420

est localement beaucoup plus faible ce qui engendre des zones non couvertes dans le nuage de points (fig. 2.7). Les 20% de surface non couverte correspondent soient à des zones de très faible incidence, soit à des parties de la paroi masquées depuis le point de vue laser par une identation. Les nuages de points bruts contiennent respectivement 5 664 657 points pour le point de vue sud et 10 479 155 pour le point de vue nord, dont respectivement 600 000 et 3 600 000 sont situés dans la zone d'intérêt ou à proximité immédiate. Des cibles réfléchissantes ont été posées en tête de paroi et sur les profils géoradar. Afin de géoréférencer les scannerisations laser et les profils radar, la position de ces cibles a été mesurées à l'aide d'une station totale.

2.1.2.4 Études géophysique

En complément de l'étude morphologique de la surface, des études géophysiques ont été réalisées par une équipe du LGIT dans le cadre de la thèse de Deparis (2007) pour détecter les principales discontinuités à l'intérieur du massif. Cette investigation comporte des profils radar en falaise, en surface et dans la fissure ainsi que des mesures électriques et électromagnétiques en surface. La figure 2.8 indique l'emplacement des différents profils dont la nature est donnée dans le tableau 2.3.

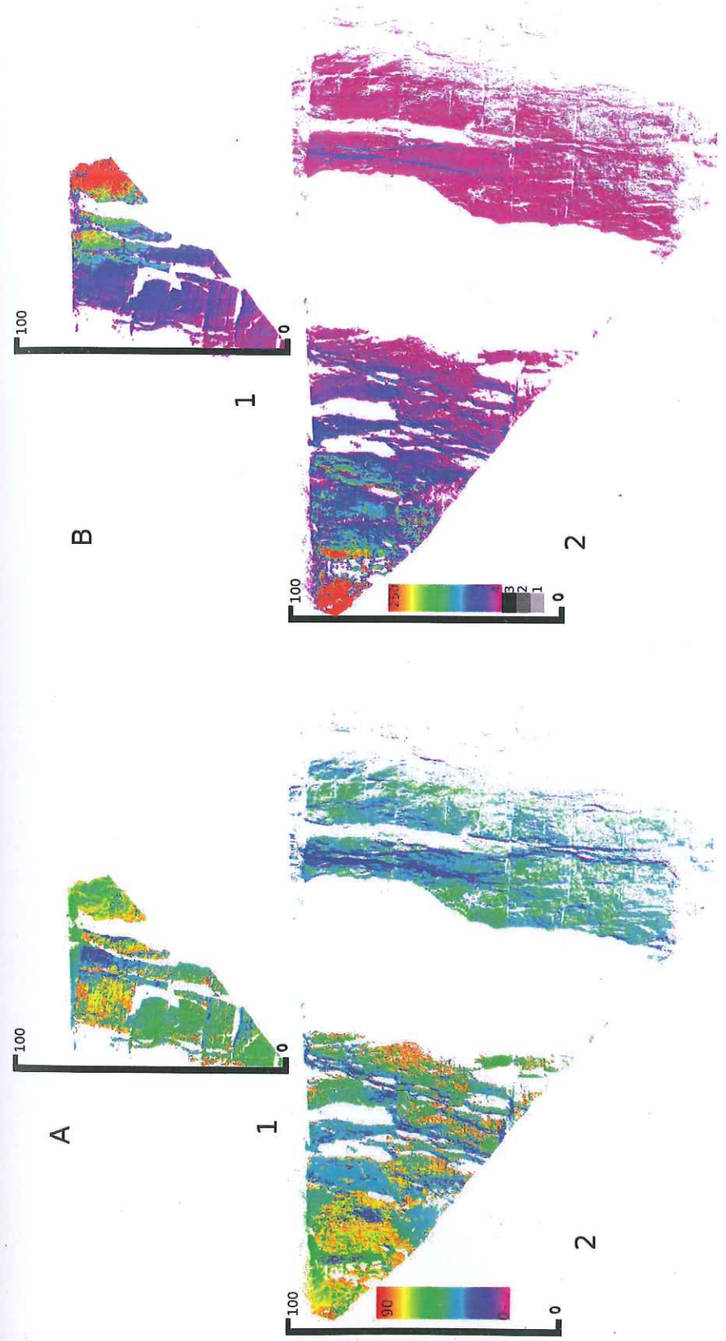


Fig. 2.7 – Influence de l'angle d'incidence sur la densité de la scannerisation laser. *Données des scannerisations lasers depuis les points de vue sud (1) et nord (2) du rocher du midi vues en perspective parallèle, perpendiculairement au plan moyen de la paroi. A gauche (A) la couleur de chaque carré de 20 × 20cm est fonction de l'angle entre le rayon laser et la paroi. A droite (B) la couleur est fonction du nombre de points lasers sur chaque carré de 20 × 20cm. L'orientation locale de la paroi est calculée à partir de 4 points dans le carré de 20 × 20cm. On observe une corrélation forte entre l'incidence locale et le nombre de points.*

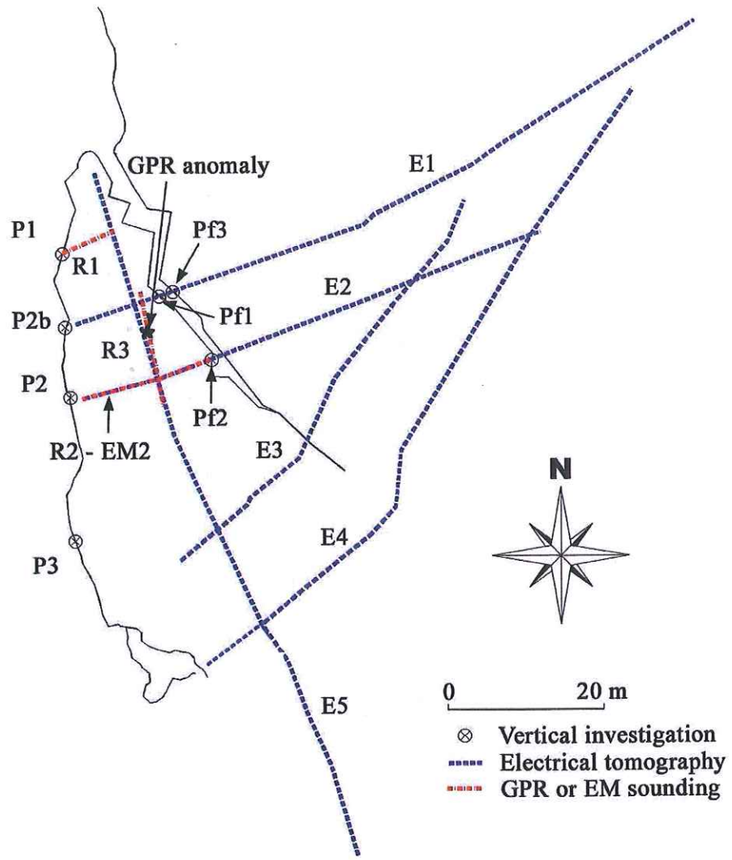


Fig. 2.8 – Plan d'implantation des profils géophysiques.
Les détails des différents profils sont données dans le tableau 2.3.

Type de profil	Nom du profil	Longueur (m)	Espacement inter-électrode (m) Espacement inter-trace (m) Mode radar – fréquence (MHZ)
acquisition sur le plateau			
Électrique	E1	94	2
	E2	63	1
	E3	62	2
	E4	94	2
	E5	94	2
EM	EM2	14.75	0.25
Radar	R1	5.40	TE – 100
	R2	15.20	TE – 100
	R2	15.20	TE – 250
	R3	12.80	TE – 100
	R3	12.80	TE — 250
acquisition en paroi			
Radar	P1	65,6	TE – 100
	P1	64,6	TE – 200
	P1	61,7	TM – 200
	P2	45	TE – 100
	P2	43,2	TE – 200
	P2b	31	TE – 100
	P3	52,4	TE – 100
acquisition dans la fissure			
Radar	PF1	54,25	TE – 250
	PF2	39,25	TE – 250
	PF3	35	TE – 250

Tab. 2.3 – Données géophysiques

2.2 Le Ravin de l'Aiguille

2.2.1 Contexte géologique

Le Ravin de l'Aiguille est situé sur le rebord Est du massif de la Chartreuse, au NE de Grenoble, au niveau de la falaise du Saint-Eynard. (fig. 2.1 et 2.9). Le versant du Saint-Eynard, très régulier, est constitué par 2 niveaux de falaises séparées par une petite vire intermédiaire raide et boisée. Ce versant s'interrompt brutalement au niveau de l'entaille très marquée et escarpée du Ravin de l'Aiguille. A ce niveau, la vire boisée

intermédiaire disparaît totalement. Le site étudié, situé au sommet de la falaise, est un tétraèdre pointe en bas dont la face visible mesure 90m de large au sommet et 150m de haut (figure 2.10). La falaise dans son ensemble mesure à cet endroit environ 800m de haut.

Du point de vue géologique, la falaise est constituée de calcaires du Tithonique présentant un pendage de 45° vers le NW, donc à contre pente. Des contrastes lithologiques sont à noter sur la hauteur de la falaise ; du haut en bas, on trouve :

- sur les 40 premiers mètres, des calcaires en bancs massifs (noté T1 à T3 sur la figure 2.10) d'une dizaine de mètres séparés par des lits marneux de 50cm environs.
- sur les 110m suivants, des calcaires marneux bien lités, en bancs d'épaisseur métrique (I1 sur la figure 2.10 puis I2 non figuré).
- au niveau de la vire intermédiaire, qui disparaît à la hauteur du ravin de l'aiguille, des calcaires marneux de 100m d'épaisseur.
- au niveau de la falaise inférieure, des dépôts marneux puis marno-calcaires sur 500m d'épaisseur.

2.2.2 Données acquises sur le terrain

2.2.2.1 Relevé des fractures en surface

Plusieurs fissures ouvertes émergent sur le plateau. Présentant des ouvertures métriques au niveau du sommet de la falaise, elles se referment en 10 à 20 mètres, et restent encore perceptibles par intermittence pendant quelques dizaines de mètres. Ces fractures ne sont pas superficielles puisqu'elles sont aussi visibles dans la falaise sur une hauteur de plusieurs dizaines de mètres. Un relevé exhaustif de ces fissures met en

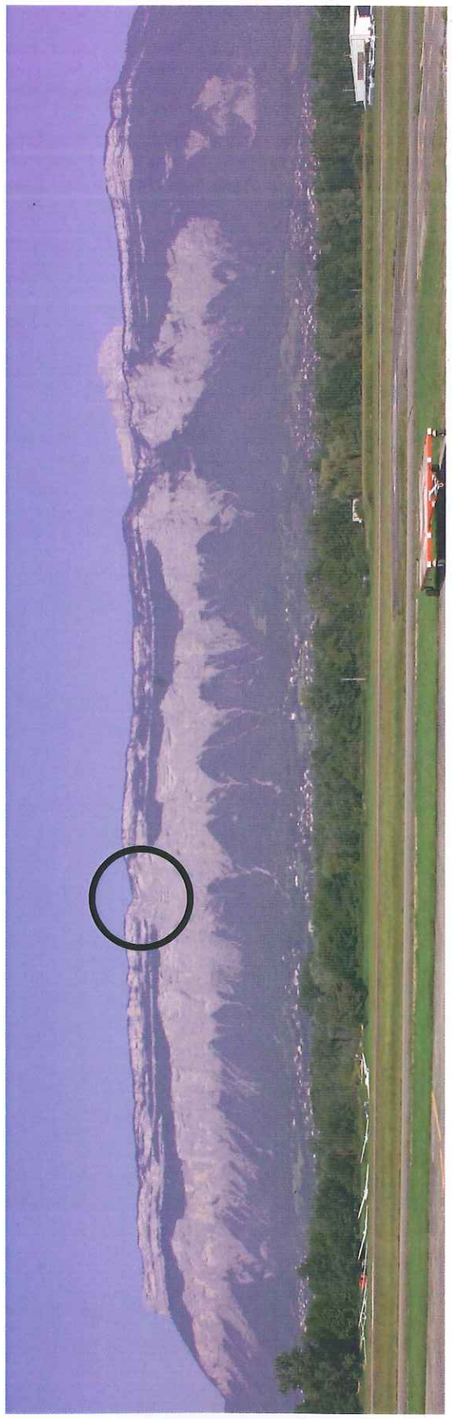


Fig. 2.9 – Vue générale de la falaise du Saint-Eynard.
La vire intermédiaire boisée disparait totalement au niveau du Ravin de l'Aiguille (entouré en noir)

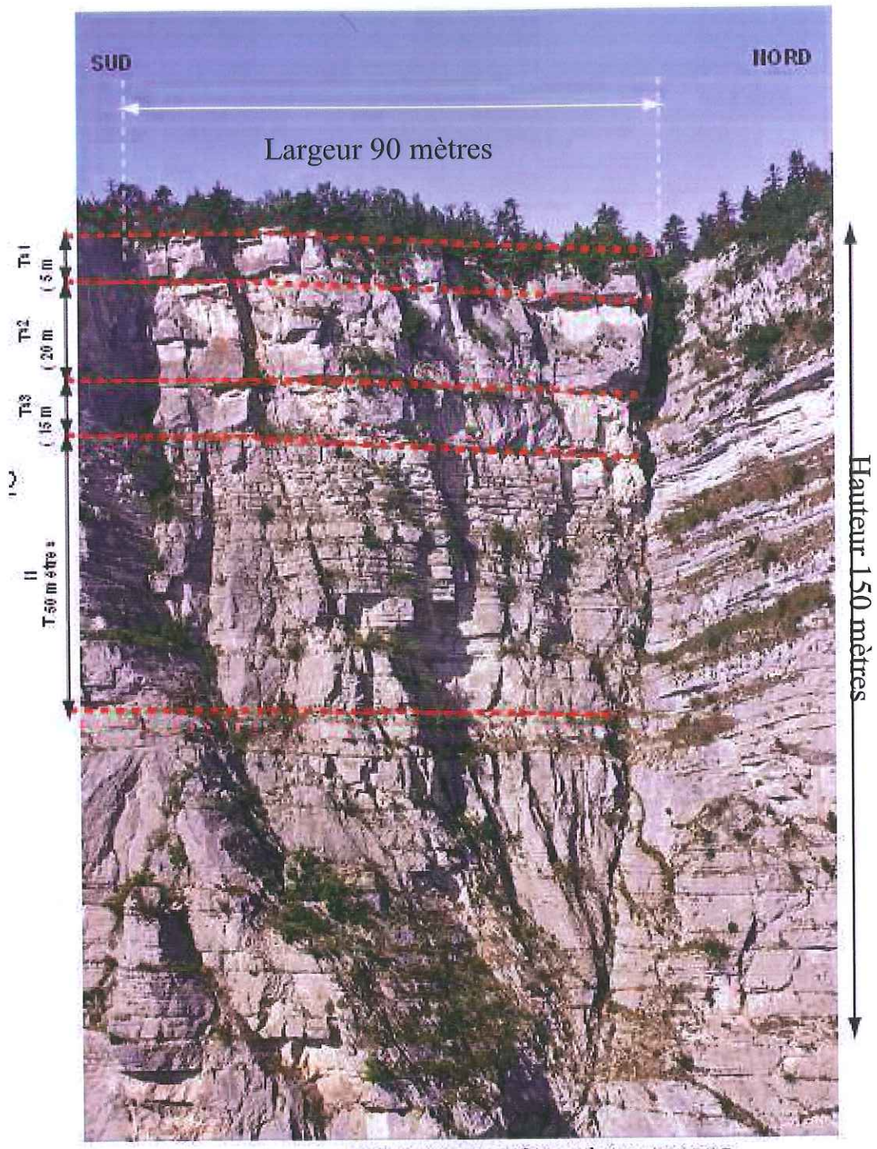


Fig. 2.10 – Vue générale du Ravin de l'Aiguille.
Les joints de stratification sont notés en rouge. La face extérieure de la zone supposée instable est le grand triangle pointe en bas de 90m par 150m. Il interrompt la régularité de la falaise.

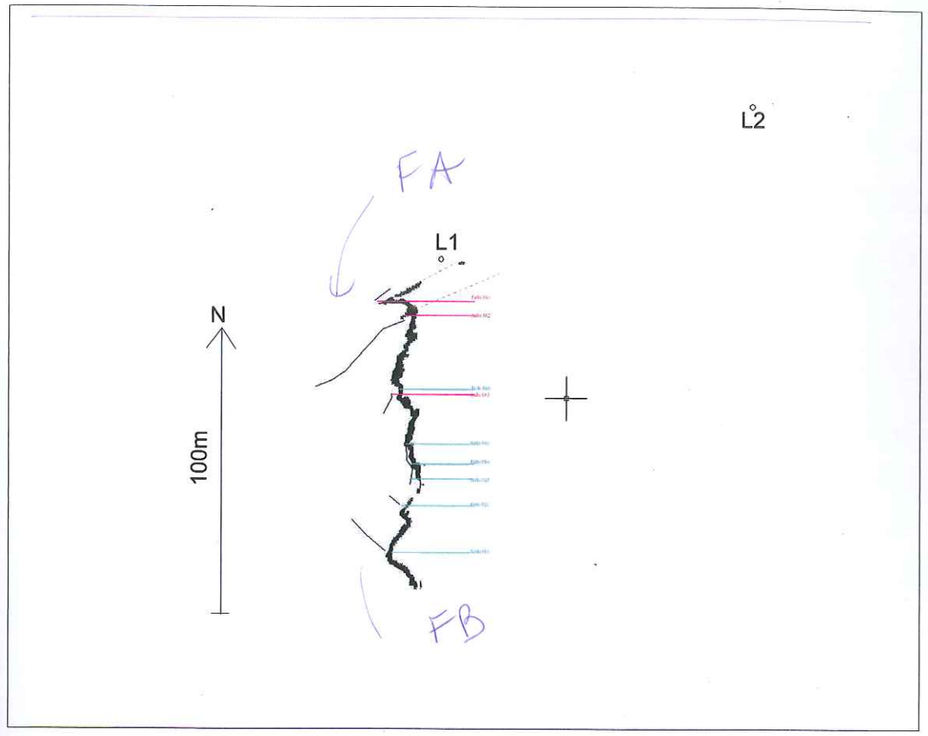


Fig. 2.11 – Plan du ravin de l'aiguille avec le relevé des principales fractures.
La limite de la paroi est issue des données de scannerisation laser. La position des fractures est établie par levé topographique. Les deux positions d'acquisition laser sont notés L1 et L2

évidence deux groupes différents *FA* et *FB* [(Jongmans *et al.*, 2007)] (fig 2.11).

La fissure principale *FA1*, qui correspond au prolongement de la falaise sur le bord droit du dièdre (fig. 2.11), présente une ouverture en tête de falaise de 1,7m et une ouverture de 20cm en bas du dièdre. Sur le plateau sa trace est visible de manière discontinue jusqu'à une soixantaine de mètres en arrière de la paroi.

Le groupe *Fa* est formé par les fissures *Fa1*, *Fa2nord*, *Fa2sud* et *Fa3*. Les fissures *Fa1* et *Fa2nord* sont d'orientation N70°E à N80°E avec un pendage de 80° à 90° vers le SE. La fissure *Fa2sud* correspond à une succession de 3 fractures en relais N70°E,

N45°E, et de nouveau N70°E, verticales ou fortement pentées vers le SE. La fissure *Fa3* est N35°E, verticale. Le groupe *Fb* est formée des fissures *Fb1* à *Fb6* qui sont orientées N130°E à N140°E verticales. Elles sont moins ouvertes que les fissures *Fa*. Elles ne présentent qu'une ouverture décimétrique au bord de la falaise, et ne peuvent être suivies que sur un peu moins de 10m. La fissure *Fa3* ainsi que le relais N45°E de la fissure *Fa2sud* pourraient être classées dans une troisième catégorie.

Le relevé exhaustif des grandes fractures a été complété par un relevé de petites fractures observables en tête de paroi. Il n'a pas été possible d'effectuer des descentes en rappel sur le dièdre pour des raisons de sécurité, et il n'y a pas d'affleurement sur le plateau à proximité de la paroi, ce qui explique le nombre particulièrement restreint de fractures mesurables (14 seulement). On retrouve cependant les deux familles *Fa* et *Fb* auxquelles peuvent se rattachent les fractures majeures : La famille *Fa* orientée N55°E à N70°E avec un pendage de 70° vers le SE et la famille *Fb* orientée N130°E avec un pendage de 80° vers le NE. La stratigraphie n'a été mesurée qu'en une seule occasion avec une direction N30°E et un pendage de 25° vers l'O

2.2.2.2 Couverture photographique

44 photographies, dont celle servant de support à la figure 2.12, ont été prises lors d'un survol en hélicoptère. L'appareil photographique utilisé était un boîtier numérique Kodak DCS 14 Pro (résolution 4500 × 3000 pixels) muni d'un objectif grand angle 24mm Nikon bloqué à l'infini. L'ensemble avait été calibré auparavant au Politecnico de Turin (Tab 2.4). Deux modèles stéréoscopiques ont ensuite été construits sur la base des quatre clichés numériques offrant les meilleurs points de vue. Ils ont été orientés grâce aux huit repères disposés préalablement en falaise et dont les positions ont été déterminées à l'aide d'une station totale au sol (fig. 2.12). Les clichés ont été pris à

une distance moyenne de 80m de la paroi, un pixel ayant une dimension moyenne de 3 à 4cm. Dans ces conditions, bien que le mode paysage ait été utilisé, Le champ est légèrement trop étroit pour observer toute la largeur du dièdre sur un seul couple stéréoscopique. Deux couples seront donc utilisés. Sur les vues rapprochées, utilisées pour les couples stéréoscopiques, les photographies couvrent presque toute la hauteur du dièdre étudié mais ne couvrent pas les 500m de paroi situés sous le dièdre.

En plus de ces photographies héliportées, des photographies ont été prises en même temps et depuis le même endroit que les scannerisations lidar (cf. 2.2.2.3). 12 photographies ont été réalisées depuis le point de vue proche et 13 depuis le point de vue lointain. Ces photographies couvrent toute la largeur du dièdre. Elles ont un recouvrement vertical de 80% et couvrent toute la hauteur de la falaise, jusqu'au pied.

Paramètre	Valeur (mm)		Distance r (mm)	Distorsion δr (μm)
taille du pixel	8		9	-102.8
focale	24.257		10	-136.2
Δx_0	0.023		11	-174.2
Δx_0	-0.112		12	-216.1
K_1	$-1.61536142 \times 10^{-04}$		13	-206.92
K_2	$2.52972731 \times 10^{-07}$		14	-307.2
Distance r (mm)	Distorsion δr (μm)		Distance r (mm)	Distorsion δr (μm)
1	-.1		15	-353.0
2	-1.2		16	-396.3
3	-4.3		17	-434.4
4	-10.0		18	-464.0
5	-19.4		19	-481.5
6	-32.9		20	-482.7
7	-51.1		21	-462.8
8	-74.4		22	-416.3

Tab. 2.4 – Paramètres de calibration de l'appareil Kodak DCS 14 pro

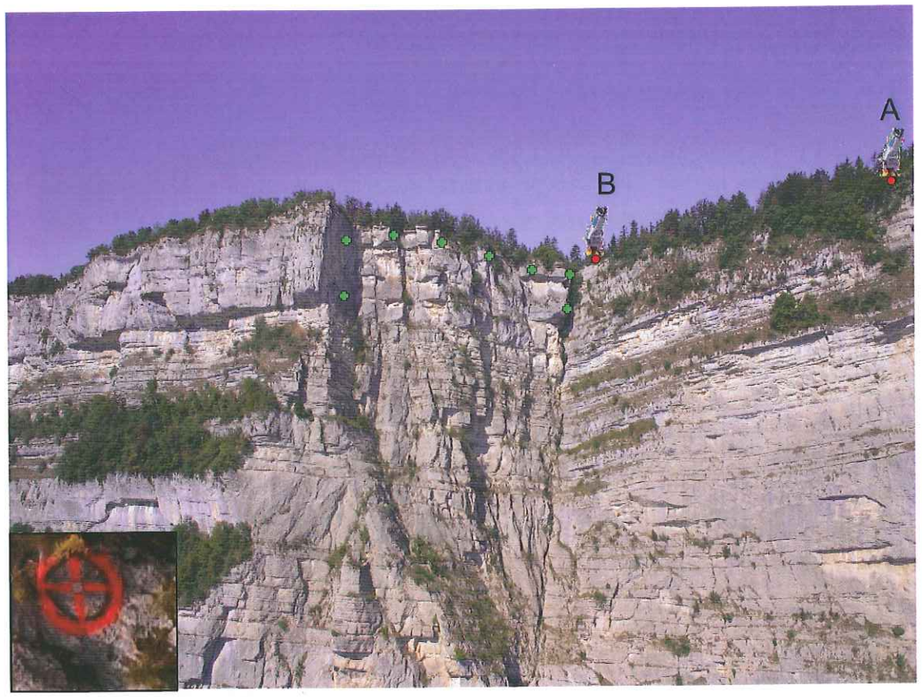


Fig. 2.12 – Position des repères de géoréférencement (croix) et des points d'acquisitions laser (A et B).
Chaque repère est formé d'une croix peinte en rose, de 50cm d'envergure environ, pour le référencement des photographies. Au centre de chaque croix une cible en matériau rétroréfléchissant a été mise en place afin de référencer les acquisitions laser (encart en bas à gauche). Une grande partie de la falaise est masquée depuis le point de vue laser le plus proche par le relief de celle-ci. Le second point de vue permet d'apprécier l'ensemble du dièdre instable.

2.2.2.3 Lidar terrestre

Des acquisitions lidar ont été réalisées depuis les deux points de vue indiqués sur la figure 2.12. Le point de vue A offre une vue d'ensemble de la falaise avec très peu de parties cachées, mais les faces orientées selon la famille *Fa* sont parallèles à l'axe du laser. Le point de vue B est plus proche de la falaise, avec pour conséquence plus de parties cachées et une vue rasante. Il permet d'acquérir les faces *Fa* de la partie nord avec un angle d'incidence important. Depuis les deux points de vue, une acquisition a été réalisée avec le scanner Riegl lms Z420i et une avec le laser Optech Ilris 3D. Pour le laser Riegl, la résolution des scannerisations était de 20 milligrades, ce qui correspond à 8 758 212 points et 11 283 862 points pour les scans A et B respectivement. Pour le laser Optech, la fenêtre de scannerisation n'étant que de $40° \times 40°$, plusieurs acquisitions ont dû être réalisées pour chaque point de vue, avec une réunion des acquisitions par reconnaissance de formes, la résolution métrique moyenne sur la paroi était de 1cm pour le premier point de vue et de 3cm pour le second. Huit cibles réfléchissantes ont été posées sur la paroi, en tête de falaise, au centre des croix photogrammétriques. Leurs positions ont été mesurées par station totale afin de géoréférencer les scannerisations laser.

2.2.2.4 Mesures géophysiques

Des mesures géophysiques ont été réalisées afin de mieux contraindre la géométrie des fractures en arrière de la paroi, en particulier pour connaître la profondeur des fractures et leur prolongement au-delà de la zone où elles sont visibles en surface [(Jongmans *et al.*, 2007)]. Le plan d'implantation des reconnaissances géophysiques est donné sur la figure 2.13. Dans un premier temps, 9 profils électriques (notées PE1 à PE9) de 155 m de long avec un espacement de 5 m entre les électrodes ont été réalisés.

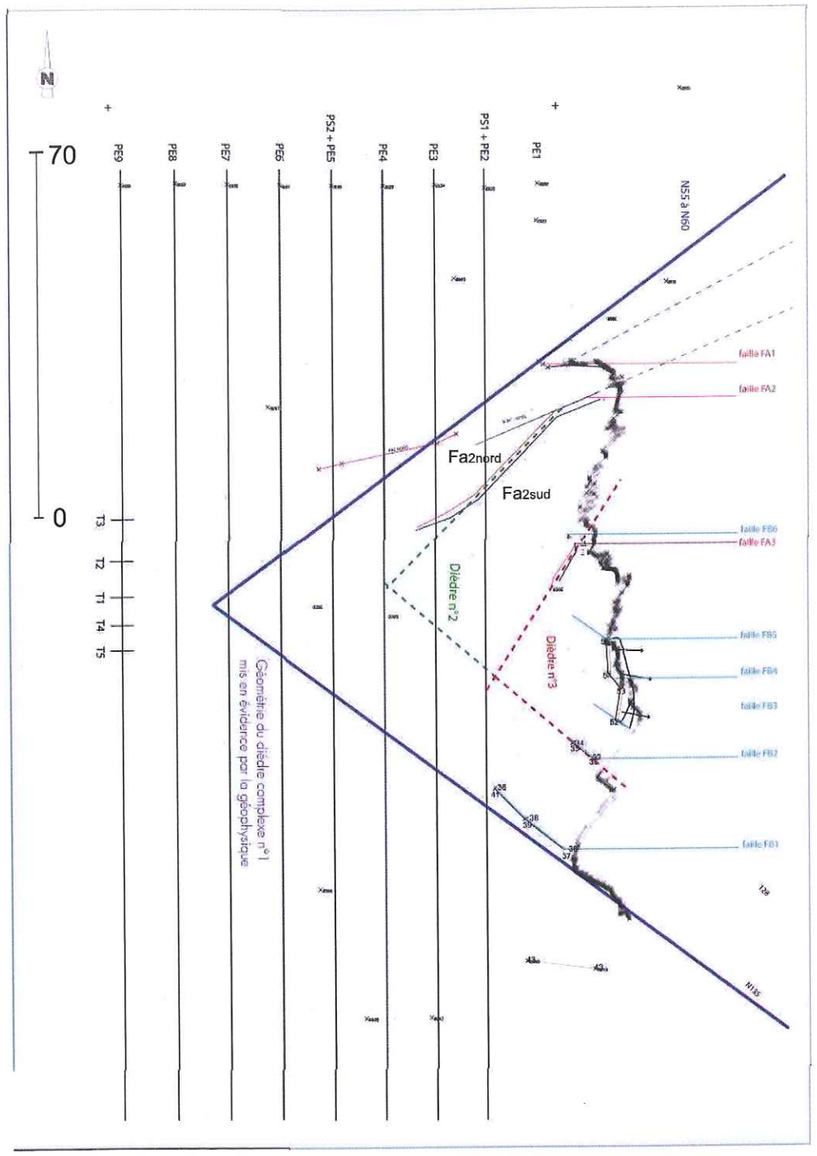


FIG. 2.13 – Plan des mesures géophysiques réalisées sur le Ravin de l'Aiguille.
Sont figurés les profils électriques PE1 à PE9, la position des tirs sismiques T1 à T5, et les trois dièdres emboitées dont la géométrie a été mise en évidence à la suite de l'étude RDT.

Les profils sont séparés de 10m les uns des autres. Dans un second temps, deux profils sismiques PS1 et PS2 de 141m de long, avec un géophone tous les 3m ont été réalisés. Les profils sismiques ont été exploités avec des tirs (émissions d'onde sismique) au niveau de chacun des géophones, avec des tirs en arrière du dièdre (tirs T1 à T5), et enfin le profil PS1 a été exploité avec une écoute du bruit sismique pendant 30 secondes.

2.3 Le Rocher de la Bourgeoise

2.3.1 Contexte géologique

Le site du Rocher de la Bourgeoise est situé sur le rebord est du Vercors (fig. 2.1). La falaise du Rocher de la Bourgeoise domine le village de St Paul de Varce. La falaise développe à cet endroit une hauteur de 300 à 400m. Elle est constituée de calcaires Urgonien massifs, d'âge Barrémien-Aptien, qui surmontent des marno-calcaires de l'Hauterivien. Le site étudié est une écaille d'un trentaine de mètres de hauteur située au sommet de la falaise (fig. 2.14).

La structure géologique de ce secteur est un grand anticlinal chevauchant vers l'ouest. Les mesures structurales [(Jongmans *et al.*, 2007)] ont permis de mettre en évidence trois familles de discontinuité : la stratification orientée N40°E/45°W et deux familles de fracture, Fa : N170°E/70°E et Fb : N50°E/70°SE. Le secteur d'étude est situé dans une zone, qui, historiquement, a connu de nombreux éboulements rocheux. Cinq éboulements concernant un volume supérieur à $1000m^3$ sont connus dans la seule commune de Saint-Paul-de-Varces depuis le $17^{ème}$ siècle [Genty, 2002]. La cicatrice supposée d'un ancien éboulement historique datée du $17^{ème}$ siècle est toujours visible sur la falaise du Rocher de la Bourgeoise. L'écaille potentiellement instable, d'environ $3000m^3$, est limitée en face supérieure et du côté sud par deux fractures ouvertes (F1

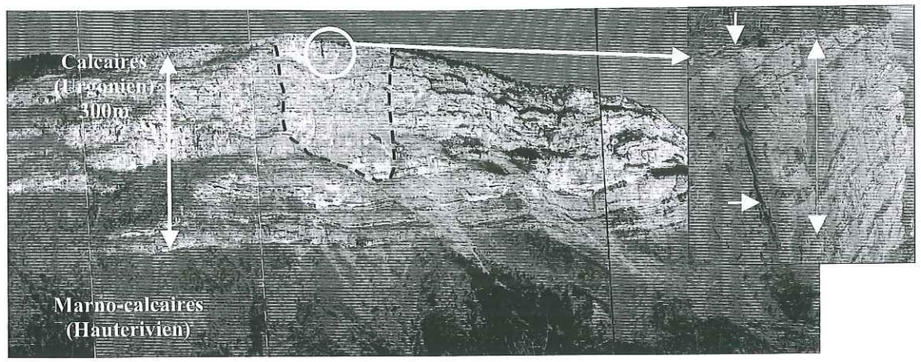


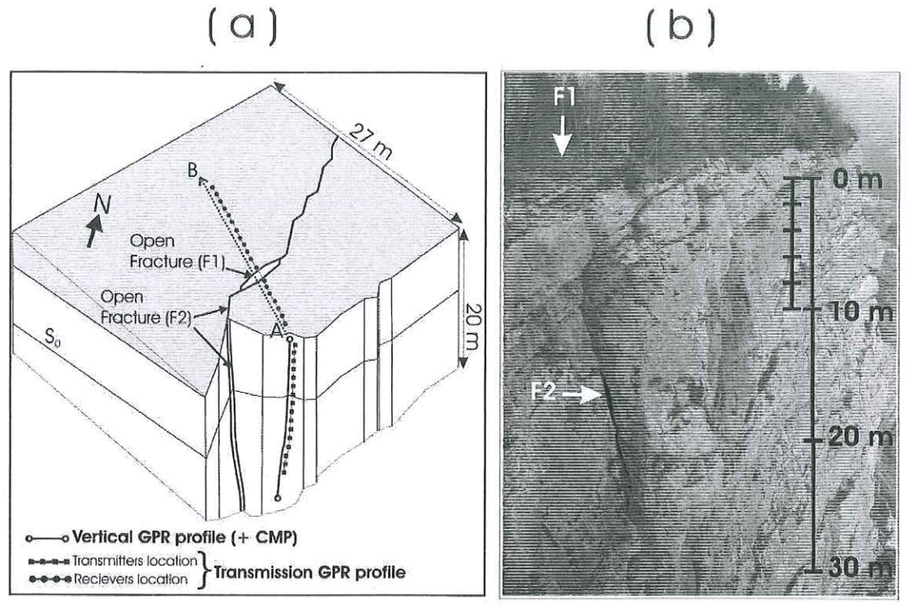
Fig. 2.14 – Panorama de la zone d'étude.
La paroi est en direction N-S, le Nord étant à droite de l'image. On distingue notamment l'émergence de la fissure principale et la limite entre Urgonien et Hauterivien [(Jongmans et al., *2007)]*

et F2) correspondant aux deux familles de discontinuités Fa et Fb (fig. 2.15).

2.3.2 Données acquises sur le terrain

2.3.2.1 Photographies et lidar héliporté

Aucun point de vue terrestre n'étant disponible pour scanner la paroi, une acquisition lidar héliporté a été réalisée. L'acquisition a été réalisée à une distance de 100m de la paroi avec un pas angulaire de 0,14°, soit un point tous les 25cm en moyenne. La précision obtenue avec un GPS cadencé à 5Hz est de 10cm sur la position du point. Des bancs de brouillard épars et mouvants étaient présents sur le siteau moment de l'aquisition. Afin d'éliminer les problèmes de masque qui en résultent, quatre passages sur le site ont été effectués. La totalité du sitereprésente un zone de 600m de large par 80m de haut. Tous les secteurs sont couverts par au moins un des passages. Il subsiste néanmoins un contraste important entre les zones couvertes par les quatre passages, où la densité de points est d'un point tous les 12,5cm en moyenne, et les zones n'ayant



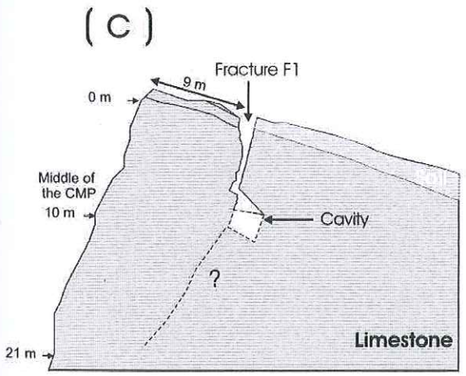
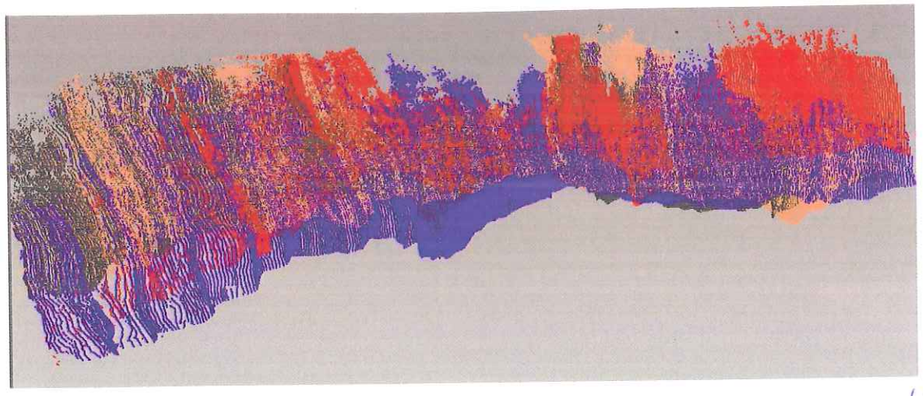


Fig. 2.15 – Site du Rocher de la Bourgeoise.
(a) Bloc diagramme du site avec la représentation des fractures F1 et F2. (b) Photographie de l'écaille étudiée. (c) Coupe de la partie supérieure de la falaise.

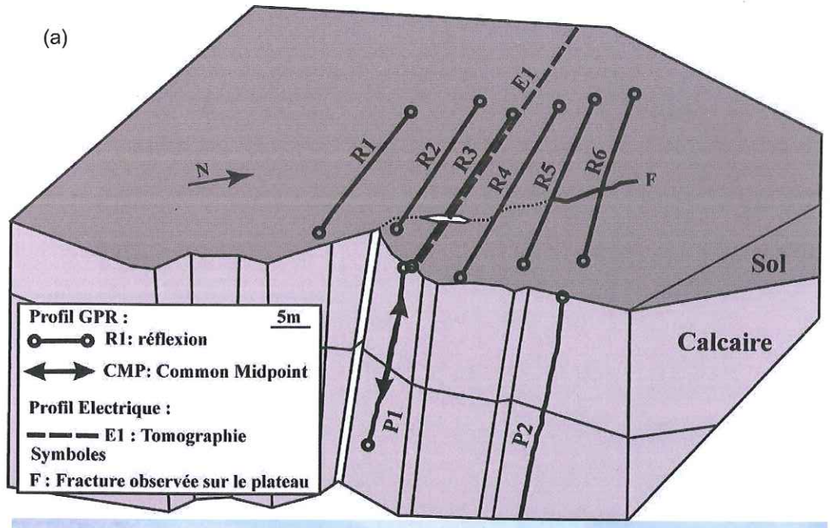


échelle ?

Fig. 2.16 – Données laser du Rocher de la Bourgeoise.
Les quatre passages sont représentés dans des couleurs différentes. On remarque de nombreuses zones monocolores ou la densité de points est bien plus faible que dans les zones où les quatre scannerisations ont pu atteindre la paroi.

pu être couvertes que sur un seul passage et qui ne contiennent qu'un point tous les 25cm (fig. 2.16).

Des photographies ont été prises en même temps que l'acquisition lidar avec un appareil numérique 23Mpixels avec une focale de 35mm. Compte tenu de la dimension physique du pixel de 9μm, le pixel moyen a une dimension sur la paroi de 3cm. Aucune stéréopréparation n'a été nécessaire pour orienter les clichés. L'appareil photographique étant, comme le scanner laser, solidaire du système IMU+GPS, l'orientation et la position des clichés sont obtenues à partir des données du GPS et de la centrale inertielle. À l'issue de cette première orientation externe, il reste cependant une erreur de parallaxe de l'ordre de 1 pixel : l'épipolaire d'un point calculé d'après le modèle ne passe pas par le point homologue réel. Cette parallaxe résiduelle est éliminée par détermination de quelques points homologues (aérotriangulation), ce qui améliore un peu la précision du modèle photogrammétrique, et surtout améliore le confort de vision du modèle stéréoscopique.



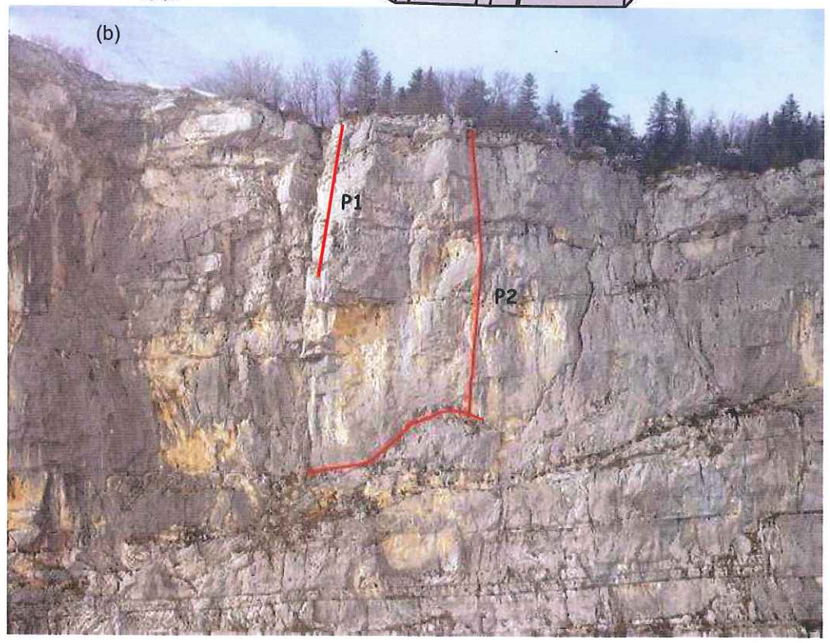


Fig. 2.17 – Implantation des profils géophysiques sur le rocher de la bourgeoise.
(a) schéma des profils sur le plateau. (b) localisation des profils en paroi sur une des photographie prise lors de l'acquisition des données laser [(Jongmans et al.*, 2007)].*

2.3.2.2 Mesures géophysiques

Les mesures géophysiques sur le Rocher de la Bourgeoise ont commencé dans le cadre de la thèse de M. Jeannin (2005). Elles ont été complétées dans le cadre du

projet CAMUS par Deparis (2007). Six profils radar et une tomographie électrique ont été réalisés sur le plateau, leurs positions respectives sont notées sur la figure 2.17 a. Les profils radar font une longueur de 20 à 30m avec enregistrement des traces tous les 20cm. L'antenne utilisée est une antenne blindée à 250 MHz. Le profil électrique, réalisé au même endroit que le profil radar P3, fait une longueur de 48m avec un espacement entre les électrodes de 50cm.

Deux profils radar P1 et P2 ont été réalisés en paroi aux emplacement indiqués sur la figure 2.17. P1 a été réalisé avec des antennes de 100, 200 et 400 MHz tandis que P2 a été effectué uniquement avec une antenne de 100 MHz. L'enregistrement des traces est réalisé tous les 20cm.

2.4 Les gorges de Paganin

La Roya est une rivière qui prend sa source au col de Tende puis parcourt 60km dans les Alpes-Maritimes et en Italie avant de rejoindre la Méditerranée (fig. 2.1). Sa vallée est orientée N-S et traverse des sédiments principalement calcaires compris entre le Trias et le Crétacé. La rivière a creusé des gorges importantes dans ces terrains, dont les gorges de Paganin, situées juste au sud de Saint-Dalmas de Tende.

Une importante route touristique, reliant Nice au nord de l'Italie, suit la vallée de la Roya jusqu'au col de Tende. Des chutes de pierres ou de blocs, de volume plus ou moins important coupent régulièrement la circulation, justifiant une étude plus détaillée des Gorges de Paganin, préalable à des travaux de mise en sécurité. Le peu de recul possible dans les gorges ainsi que la difficulté d'accès au sommet de celles-ci rend cependant difficile le choix de points de vue pertinents. L'étude de mise en sécurité réalisée par IMSRN a été l'occasion d'une collaboration entre IMSRN et le LGCA, IMSRN testant

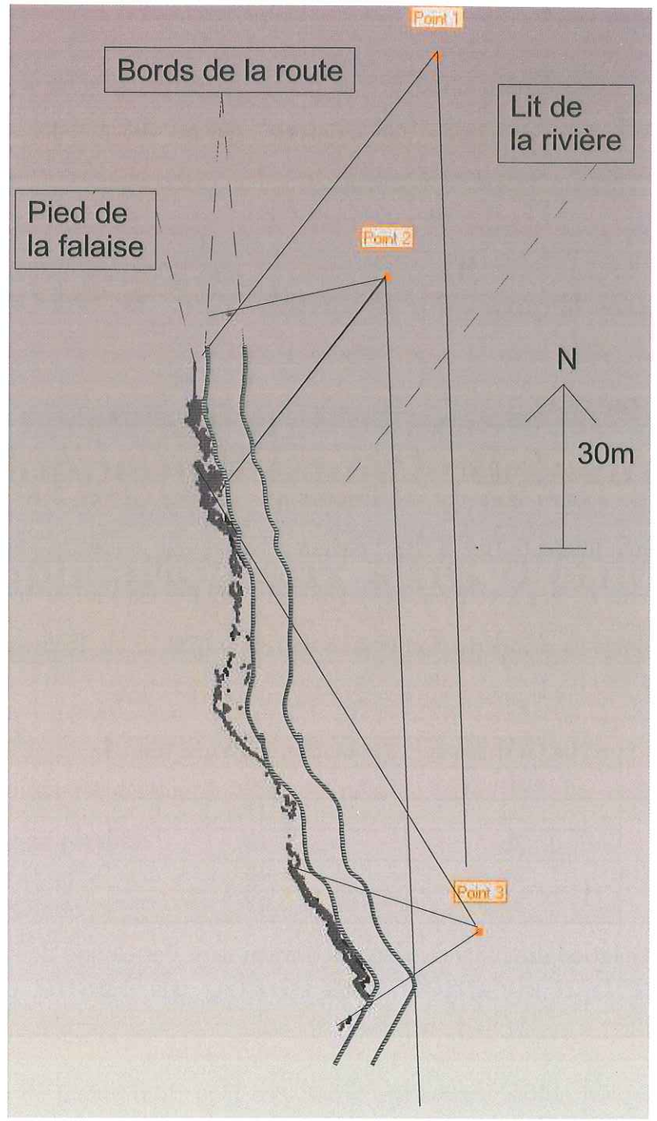


Fig. 2.18 – Plan du site de Paganin.
Le tracé du bord de la route au plus loin de la paroi est approximatif. Les trois points d'acquisition laser sont figurés avec la zone couverte par chaque scannerisation. À cause de la faible ouverture angulaire du scanner Optech, deux acquisitions ont été nécéssaires pour les points de vue 2 et 3. Aucun point de vue plus éloigné n'était disponible, la falaise de l'autre rive de la Roya commençant immédiatement après les points de vue utilisés. Le tracé du pied de la falaise est obtenu en réalisant une tranche dans le bas du nuage de points.

le potentiel d'une scanerisation laser sur une partie des gorges sans en supporter la totalité du coût, le LGCA réalisant une acquisition sur un site instable, posant la problématique des points de vue en contrebas et non en contrehaut.

2.4.1 Données acquises sur le terrain

2.4.1.1 Lidar terrestre

Les données de scanner laser ont été acquises avec le laser Optech Ilris 3D depuis trois points de vue différents (fig. 2.18). Le champ balayé par le laser n'étant que de 40 degrés, les points de vue 2 et 3 ont été acquis en deux fois sans recouvrement. Les résolutions moyennes de scannerisation étaient de 18mm (Tab. 2.5). Trois sphères par acquisition ont été utilisées pour géoréférencer les nuages de points.

n° de l'acquisition	Résolution moyenne(mm)
1	18
2a	17
2b	19
3a	18
3b	17

Tab. 2.5 – Résolution des différentes scannerisation laser des Gorges de Paganin

2.5 Conclusion

La qualité des points de vue étaient très variable d'un site à l'autre parmi les sites étudiés, imposant de recourir à différente méthodes d'acquisition des données. Nous allons voir dans le paragraphe suivant dans quelle proportion ces différences au niveau de l'acquisition ont une influence sur le traitement et l'interprétation des données.

Chapitre 3

Développements méthodologiques et Application aux sites d'étude

Dans ce chapitre, je vais décrire les méthodes que j'ai employées, en détaillant les méthodes qui ont été développées dans le cadre de cette thèse. Les résultats obtenus seront décrits en parallèle.

3.1 Du nuage de points aux données utilisables

Le nuage de points brut, qu'il soit issu d'une scannerisation laser ou d'une corrélation automatique, n'est pas pratique à exploiter en l'état. On va donc chercher à organiser l'information de façon à l'exploiter plus facilement. La méthode classique d'organiser l'information est de réaliser un modèle numérique de surface (MNS) qui organise les points bruts en une surface triangulée. Une méthode alternative se développe depuis quelques années au Politecnico de Turin (Italie), basée sur le concept d'image solide [Bornaz et Dequal, 2003].

3.1.1 Le modèle de surface

Un modèle de surface diffère d'un nuage de points par le fait que la position de la surface y est modélisée de manière continue, et non seulement de manière discrète. Dans de nombreux cas de figure, le modèle numérique de surface ou de terrain peut être exprimé sous la forme d'une grille à pas régulier suivant deux axes de coordonnées X et Y, avec la donnée de la valeur de Z à chaque nœud de la grille. La valeur de Z en tout point d'une maille est interpolée en fonction des valeurs de Z aux sommets de cette maille. Dans le cas des falaises cette approche n'est pas applicable du fait de redoublement de la surface (fig. 3.1). On modélise alors la falaise par un ensemble de polygones, généralement des triangles, dont toutes les faces sont supposées correspondre à la surface réelle.

L'étape de création proprement dite du modèle de surface est l'étape de triangulation énoncée en 1.3.2.4. En plus de cette étape, les autres opérations décrites en 1.3.2, à savoir géoréférencement, fusion des différentes acquisition, élimination des points aberrants et réduction du bruit, doivent être effectuées. Suivant le nombre d'acquisitions laser et leur taux de recouvrement, l'ordre des opérations et la méthode utilisée pour certaines des opérations varieront légèrement.

3.1.1.1 Géoréférencement

Nous considérons ici le référencement des différents nuages de points.

Le référencement ne se pose pas dans les mêmes termes selon que l'on traite un nuage isolé ou plusieurs nuages de points. Avec un seul nuage de points le problème se résume à l'ajustement d'un seul faisceau et à l'estimation de la translation/rotation permettant de passer du système de coordonnées laser au système de coordonnées terrain. Avec

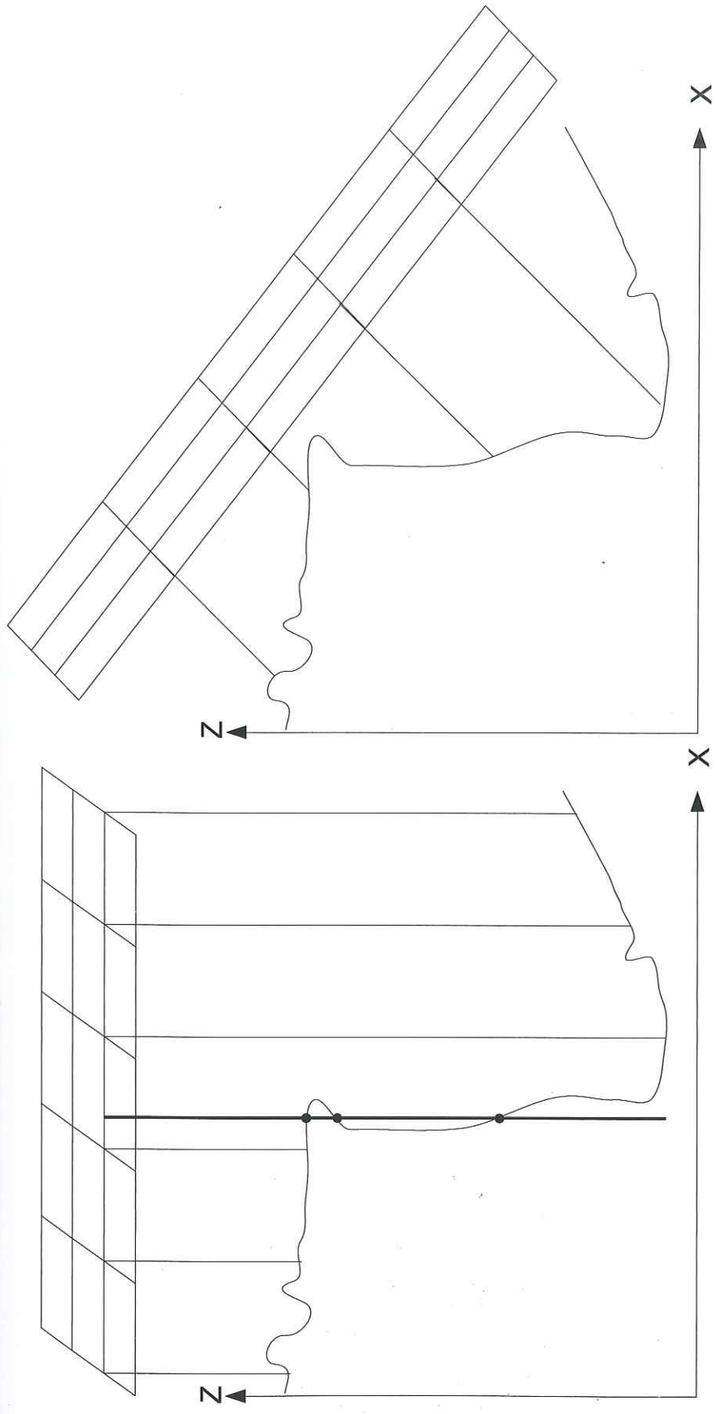


Fig. 3.1 – Limite des modèles numériques de surface exprimés sous forme de grille. *À gauche, un MNT exprimé sous forme de grille. Pour chaque nœud de la maille en (X, Y), la valeur de Z du sol est donné par le MNT. En présence de surplomb, un même nœud (X, Y) correspond à trois positions du sol. À gauche, ce problème peut parfois être contourné en utilisant un plan de projection parallèle au plan moyen de la paroi. Cette astuce n'est pas généralisable à toutes les surfaces, elle nécessite l'existence d'un plan de projection pour lequel chaque point du plan correspond à un unique point du sol.*

deux nuages ou plus, se chevauchant en partie, l'ajustement vise à obtenir un modèle global qui minimise, sur l'ensemble, les écarts entre le modèle et les mesures. Tant que cet ajustement repose sur des points, il est possible d'en déterminer la précision. Cependant, dès que l'on doit faire appel à des ajustement de forme, il devient plus problématique d'estimer cette précision.

Le géoréférencement est la première opération à effectuer. Sur le site du Rocher de la Bourgeoise, l'utilisation d'un lidar héliporté a imposé un géoréférencement par GPS et IMU comme détaillé en 1.3.3. Sur les sites du Rocher du Midi et du Ravin de l'Aiguille, le référencement a été réalisé en plaçant des cibles réfléchissantes sur la paroi. Lorsque plusieurs tirs laser sont rétrodiffusés par une même cible, le calcul de la position du milieu de la cible se fait en calculant le barycentre des points obtenus, pondérés en fonction de l'intensité de la réflexion [Bornaz 2005].

L'utilisation du laser Riegl, ayant un champ d'acquisition de 360°×80°, permet de positionner des cibles en arrière de la position de scannerisation. De cette manière on réussit toujours à placer des réflecteurs dans le champ du laser avec une répartition spatiale permettant d'obtenir une bonne précision sur l'orientation du système laser.

Sur le site du Ravin de l'Aiguille, une série d'acquisitions a été réalisée avec le scanner Optech en plus de celles réalisées avec le scanner Riegl. À cause de la faible ouverture angulaire du laser Optech, il n'était pas possible de couvrir la falaise du haut en bas en une seule acquisition (fig. 3.2). Comme les cibles réfléchissantes se trouvaient toutes au sommet de la falaise, l'acquisition du pied de la falaise ne pouvait donc pas être référencée indépendamment et ne pouvait l'être que par reconnaissance de forme. Dans ce cas de figure, il faut assurer un recouvrement suffisant entre les différents nuages pour que la reconnaissance de forme converge rapidement et soit suffisamment précise. Typiquement, un recouvrement de 25 à 30% est recommandé pour des résultats

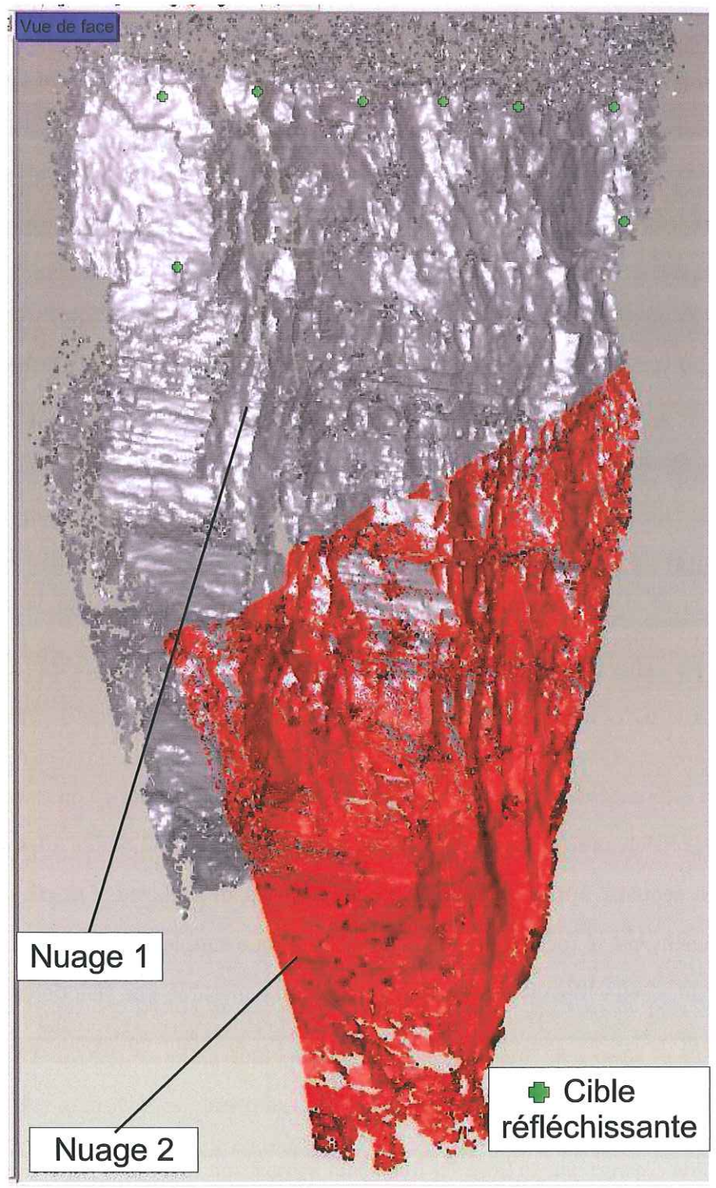


Fig. 3.2 – Acquisition laser Optech sur le site du Ravin de l'aiguille.
Les points noirs et les points rouges appartiennent à deux acquisitions successives depuis le même point de vue. Les deux scannerisations se recouvrent en partie.

corrects [Rabbani *et al.*, 2007]

Sur le site de Paganin, le géoréférencement a été réalisé à l'aide de sphères. Le laser utilisé était le laser Optech. Une scannerisation supplémentaire est réalisée sur chaque sphère à la résolution maximum du laser. Du fait de la faible ouverture angulaire du laser (40°×40°) et de la configuration du site défavorable, les sphères étaient trop proches et trop alignées si bien que la précision obtenue n'a pas suffi pour réunir les différents points de vue. L'écart résiduel au sommet de la falaise entre les différents nuages de points était d'environ 1m, soit 0,5° d'erreur sur l'orientation de l'axe vertical. Un référencement par reconnaissance de forme a donc été utilisé pour améliorer le modèle général. La reconnaissance de forme n'est réalisée qu'après avoir éliminé les points aberrants. La végétation est conservée à ce stade pour deux raisons : d'abord la forme de celle-ci participe à faire converger la reconnaissance de forme, ensuite il est plus facile de l'éliminer après avoir réuni les nuages de points, lorsque les masques qu'elle crée sur la paroi sont éliminés par la réunion des différents points de vue.

Au final, le référencement par sphère s'avère moins pratique à mettre en œuvre que celui par cibles réfléchissantes. Les sphères masquent la paroi, ce qui oblige à les mettre en place et les acquérir après ou avant la scannerisation de la paroi. Les cibles sont collées directement sur le rocher, ne créant pas de masque supplémentaire. De plus, la position des cibles réfléchissantes peut être directement mesurée par théodolite, alors qu'une sphère doit être remplacée par un prisme posé sur la même embase avant d'être mesurée.

L'angle solide couvert par le lidar en une seule acquisition est une donnée limitante fondamentale. Un laser ayant un petit champ d'acquisition oblige à disposer les cibles servant au référencement dans une zone relativement réduite, dans laquelle il peut être difficile de placer des cibles suffisamment dispersées. Il faudra de plus disposer d'un

moyen de géoréférencer chacune des acquisitions indépendantes nécessaires pour couvrir toute la falaise. Lorsqu'il n'est pas possible de poser de cible sur une partie de la paroi et que l'on est contraint d'utiliser la reconnaissance de forme pour le référencement, il est nécessaire de scanner deux fois une part importante de la paroi pour que la reconnaissance de forme donne un résultat.

Pour toutes ces raisons, un laser ayant un champ de couverture le plus grand possible est non seulement plus pratique sur le terrain en ne nécessitant pas de changer manuellement l'orientation du laser entre deux acquisitions depuis le même endroit, mais surtout plus précis en évitant les erreurs liées au référencement par mesure de forme et en permettant de placer bien plus facilement des cibles réfléchissantes loin les unes des autres. Les erreurs dûes au référencement par reconnaissance de forme sont d'autant plus gênantes qu'elles ne sont pas quantifiables dans la forme originelle de l'algorithme, et ce même si des travaux ont été fait pour réduire cette limitation [Guering, 2001].

3.1.1.2 Élimination des points aberrants, réduction du bruit et élimination de la végétation

L'élimination des points aberrants et la réduction du bruit n'ont pas posé de difficultés particulières, les algorithme classiques s'appliquant bien dans le cas des paroi rocheuse. Des algorithmes classiques, basés sur le filtre médian, ont été utilisés. L'élimination de la végétation est une opération plus délicate. Une élimination manuelle peut être très fastidieuse. Les algorithmes décris en 1.3.2.2 peuvent être utilisés pour éliminer la végétation. Les résultats seront relativement bons lorsque les points correspondants à la végétation seront bien séparés de ceux correspondant au sol et que le sol est atteint suffisamment souvent pour que sa position puisse être raisonnablement

extrapolée partout. Ces hypothèses sont vérifiées pour de la végétation haute et peu dense, mais moins bien pour de la végétation basse et dense. Suivant la sensibilité de l'algorithme et la proportion de points atteignant le sol, on risque soit d'éliminer des points appartenant à une saillie rocheuse soit de ne pas éliminer un arbuste suffisamment dense.

Les acquisitions des trois sites grenoblois ont été réalisé en automne et au début du pritemps pour tenter de limiter la couverture végétale tout en évitant la période d'enneigement. Un second moyen de s'affranchir de la végétation est de travailler au maximum sur l'image solide (3.1.2) plutôt que sur le nuage de points ou le modèle de surface. La végétation est facilement identifiable sur l'image solide. Il suffit de ne pas réaliser de mesures là où de la végétation est visible pour que les mesures n'en soient pas perturbées. Cette logique est aussi applicable sur un modèle de surface texturé avec précision. Toutes les mesures locales peuvent être réalisées sur des zones non couverte par la végétation. Seules les opérations automatiques et l'utilisation du modèle de surface pour des simulations numériques (3.5.3) imposent une élimination de la végétation.

3.1.1.3 Triangulation et réunion des nuages de points.

Les deux opérations restant à effectuer sont la triangulation du nuage de points et la réunion des données provenant de différentes acquisitions. L'ordre dans lequel on pratique ces deux opérations a une importance cruciale. En effet, la réunion de deux ou plusieurs nuages de points est une opération très simple : il suffit de rassembler dans un seul nuage les données provenant des deux nuages. Le rattachement de deux surfaces triangulées est au contraire une opération complexe. Sur les zones communes à deux triangulations, on dispose de deux interpolations différentes de la surface. La surface

« moyenne » de ces deux interpolations est celle qui minimise les carrés des distances entre elle-même et les deux autres interpolations. Le calcul de cette distance est déjà en soit une opération coûteuse en temps de calcul et en mémoire, la détermination de la surface qui minimise cette distance l'est nécessairement encore plus. Parallèlement à la difficulté de réunir deux surfaces triangulées, la triangulation est une opération complexe, comme nous l'avons évoqué en 1.3.2.4. La quantité de mémoire nécessaire est proportionnelle au nombre de points et le temps de calcul est proportionnel au carré du nombre de points. Si l'on souhaite trianguler tous les points issus de la réunion de deux nuages de points, il faut donc deux fois plus de mémoire et quatre fois plus de temps avec le même algorithme que pour réaliser les triangulations séparément.

Dans le cas d'un nuage de points issu d'une seul scannerisation laser, la triangulation peut être grandement simplifiée. Les points issus du scannerisation laser peuvent être exprimés en géométrie sphérique selon des coordonnées (R, θ, ϕ) dans le référentiel laser. L'intérieur du massif se trouve forcément du coté des R supérieurs aux points obtenus et le vide du coté des R inférieurs aux points mesurés. La figure 3.3 nous montre que l'interpolation correcte des points proches sur la surface ne dépend que de θ et ϕ,une interpolation de la surface à partir de points proche en R mais distant en θ et ϕ ne permettant pas de respecter l'orientation intérieur/extérieur du massif imposée. La triangulation donnant le résultat le plus proche de la surface réelle est donc la triangulation faite en deux dimension sur les coordonnées θ et ϕ. Si la triangulation de Delaunay est équivoque en trois dimensions, elle est univoque en deux dimensions et n'est pas plus compliquée qu'un tri des points selon les deux « pseudo » directions θ et ϕ.

Le laser fonctionnant en réalisant des lignes de points avec θ fixe, et ϕ croissant puis en augmentant θ d'un incrément (ou réciproquement des lignes avec ϕ fixe et θ croissant puis en augmentant ϕ d'un incrément selon le modèle de laser), le classement

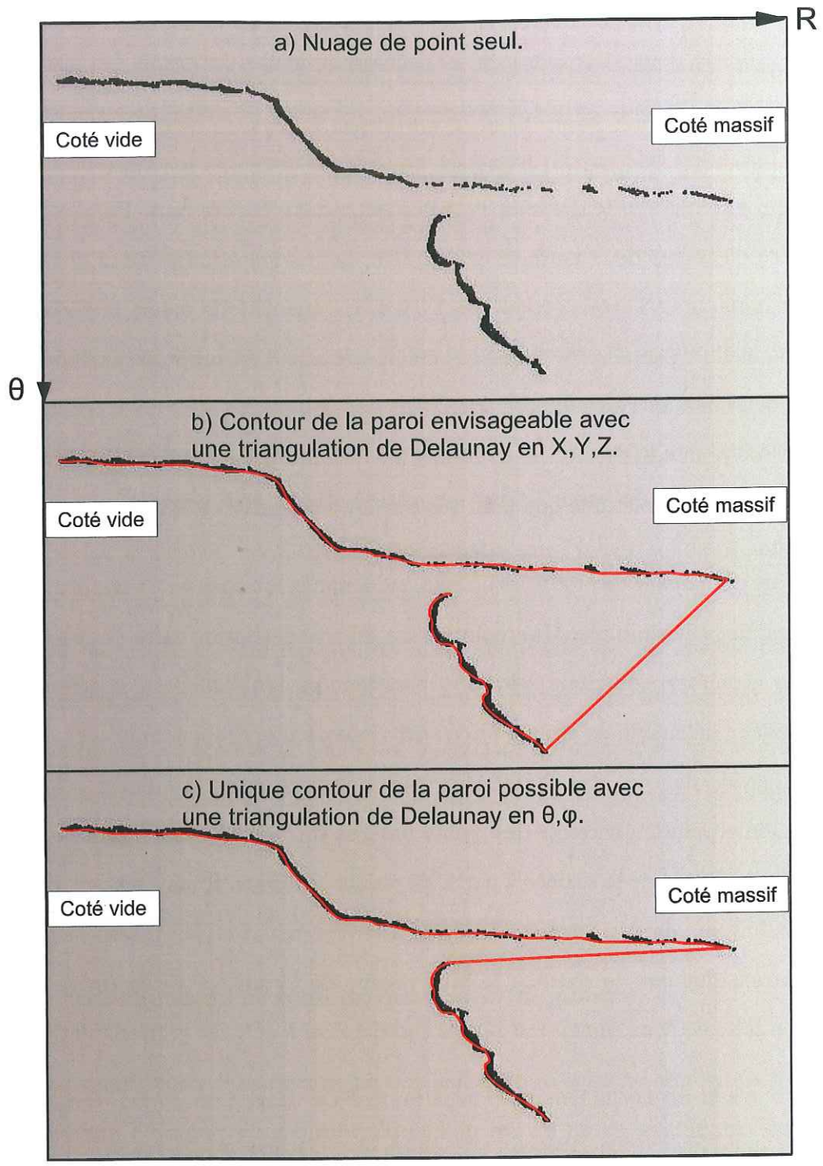


Fig. 3.3 – Triangulation en géométrie sphérique, exemple du Rocher du Midi.
La coupe est réalisée à un endroit où la fracture principale émerge au niveau de la surface de la paroi.
a) Coupe dans un plan (R, θ) d'un nuage de points laser issu d'une même scannerisation. La falaise étant toujours située du côté des R « grands » une triangulation faite comme en b) ne peut correspondre à la réalité, la triangulation correcte est celle illustrée en c).

des points en fonction de leurs coordonnées θ et ϕ est déjà réalisé dans le fichier brut issu de l'acquisition. Il faut toutefois faire attention au fait que les logiciels trient les points lors des opérations de filtrage ou de détermination des positions des cibles, et ne conservent pas l'ordre d'acquisition des points. Lorsque l'ordre d'acquisition est conservé, le temps de triangulation est linéaire en fonction du nombre de points, et le coût en mémoire est proportionnel à $\sqrt{n}$.

On a donc d'un côté la réunion des nuages de points qui est très simple et la triangulation dans le cas général complexe, de l'autre la triangulation dans la géométrie du scanner laser qui est très simple et la réunion des surfaces interpolées qui est complexe. Sur le site du Rocher du Midi, on dispose de deux scannerisations laser ayant un recouvrement presque nul. Une triangulation exploitant l'ordre des points suivie d'une réunion des deux surfaces est donc parfaitement adaptée. La petite zone d'intersection entre les deux surfaces a été traitée de manière manuelle en éliminant, sur la zone de recouvrement, les triangles les plus extérieurs au massif jusqu'à élimination de la redondance. Sur les autres sites, la réunion des acquisitions s'est faite au stade du nuage de points. Les paramètres de référencement des nuages de points comportant une erreur résiduelle, le nuage de points résultant de la réunion est bruité. Si ce bruit n'est pas réduit par un nouveau filtrage, la surface interpolée possèdera une rugosité artificielle importante.

3.1.1.4 Simplification du modèle de surface

Le passage du nuage de points au modèle de surface s'accompagne d'un accroissement de la quantité de données à stocker. En effet, on ajoute à N points ayant trois coordonnées spatiales, chacune encodée sur 32 bits ou (mais c'est plus rare) 64 bits, la définition de $2N$ triangles. Chaque sommet composant le triangle peut être mémorisé

par son numéro, encodable sur 24 bits si le nuage comporte moins de 16 millions de points. Cela représente $3 \times 32 \times N = 96N$ bits minimum pour le nuage de points et $3 \times 24 \times 2N = 144N$ bits supplémentaires pour la surface triangulée. Des définitions optimisées de la surface en utilisant des bandes de triangle permettent de réduire cette taille en ne définissant qu'un sommet par nouveau triangle, les deux autres sommets étant les deux derniers sommets du triangle précédent. Si le coût de stockage des données d'une surface triangulée est ainsi proche de celui des nuages de points, l'affichage est beaucoup plus coûteux. Afin de faciliter tant le stockage que la visualisation ou les calculs ultérieurs, on a donc souvent recours à des simplifications de la surface.

Cette simplification peut être faite au stade du nuage de points, permettant ainsi de réaliser des triangulations à partir de nuages initialement trop conséquents. À ce stade, plusieurs méthodes peuvent être appliquées. La simplification peut être faite en même temps que la réduction de bruit, ou après celle-ci. Comme les erreurs sur R, θ et ϕ sont indépendantes, il est préférable de réaliser la réduction du bruit et la simplification dans le repère sphérique lié au laser. On peut encore réduire le nombre de points après la réunion des divers nuages de points. Les algorithmes d'échantillonnage basés sur la courbure mesurent localement la courbure de la paroi et conservent d'avantage de points dans les zones où cette courbure est importante. Ils préservent ainsi d'avantage de points dans les zones anguleuses [Pauly *et al.*, 2002] et sont donc bien adaptés pour conserver au mieux les plans de fracture.

La simplification peut aussi être effectuée après la triangulation. L'intérêt de ne simplifier les données qu'après la triangulation est que la surface, ou du moins son interpolation, est connu. On peut donc éliminer les triangles là où cette simplification modifie peu la position de la surface interpolée. Deux grandes classes d'algorithmes de simplification existent, les algorithmes par suppression de sommet et ceux par abaissement d'arêtes [Cignoni *et al.*, 1998]. Les deux classes d'algorithmes fonctionnent de

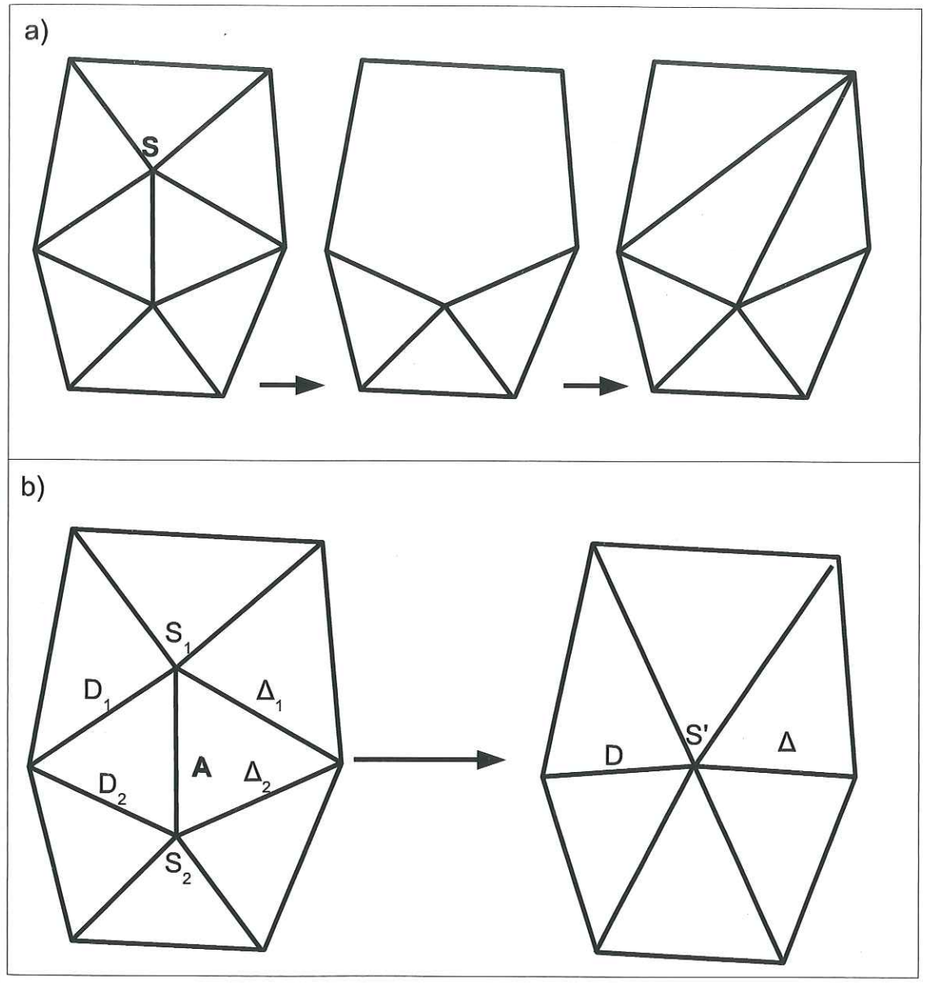


Fig. 3.4 – Simplification d'une surface triangulée.
a) par suppression du sommet S. b) par abaissement de l'arête A.

façon itérative : une opération élémentaire (suppression d'un sommet ou abaissement d'une arête) est effectuée là où la distance entre l'interpolation résultante et la surface initiale est minimale, et cette opération est répétée tant qu'une condition d'arrêt n'a pas été atteinte. Cette condition peut être soit la réduction du nombre de triangles jusqu'à un nombre fixé par l'utilisateur soit l'impossibilité de retirer un nouveau triangle sans dépasser une distance maximale admissible entre les deux surfaces.

La suppression de sommets est réalisée en supprimant un sommet S et les arêtes qui y prennent appui, puis en recréant de nouvelles arrêtes entre les sommets qui entouraient le sommet supprimé (fig. 3.4 a). Les sommets restant à la fin de l'opération sont donc situés au même endroit que des sommets de la triangulation initiale.

L'abaissement d'arête est réalisé en prenant une arête A et en réduisant sa longueur à 0 (fig. 3.4 b). Les sommets S_1 et S_2 de la figure deviennent confondus en un seul sommet S'. Les arêtes D_1 et D_2 sont confondues en D, les arêtes Δ_1 et Δ_2 sont confondues en Δ. Le sommet S' n'est pas nécessairement à la position de S_1 ou S_2, mais se trouve à l'emplacement sur l'arête initiale qui va minimiser la distance entre les surfaces avant et après abaissement de l'arête. La position des sommets restants à la fin de l'opération est donc différente, pour un certain nombre d'entre aux, de la position des sommets de la triangulation initiale.

3.1.2 Le concept d'image solide

Le concept « solid image », breveté par le Politecnico de Turin, a été présenté pour la première fois en 2003 [Bornaz et Dequal, 2003]. En collaboration avec le Politecnico de Turin, j'ai développé des outils d'exploitation de l'image solide. Ceux-ci ont été créés au fur et à mesure des besoins pour l'étude des escarpements instables, mais sont réutilisables dans des cadres très variés.

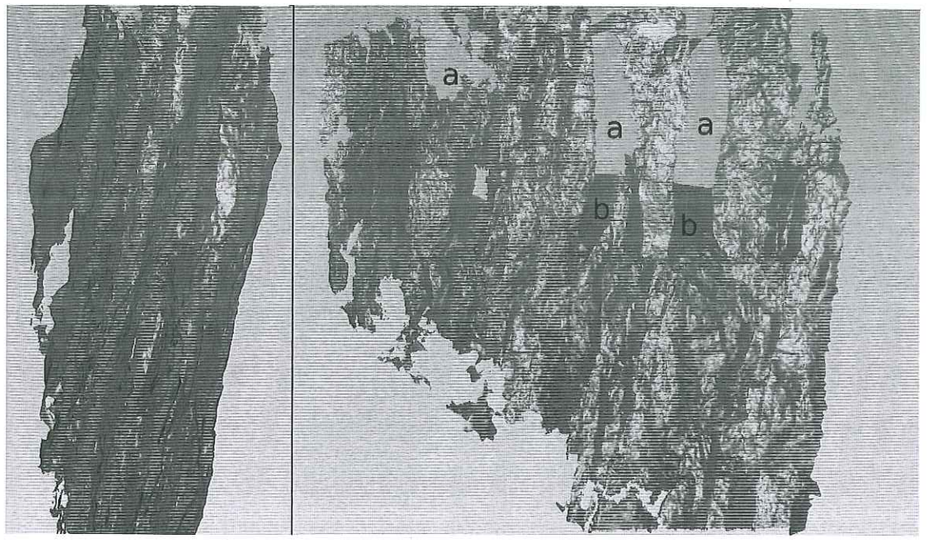


Fig. 3.5 – Visualisation du MNS du rocher du midi depuis deux points de vue différents. *À gauche depuis le point d'acquisition laser, la visualisation est correcte. À droite depuis un point de vue situé face à la paroi. Les zones peu couvertes par la scannerisation laser laissent des trous dans la paroi (a) ou forment des triangles très allongés (b)*

3.1.2.1 Définition

Lorsque l'on dispose d'un nuage de points ou d'un modèle de surface ainsi que d'une photographie, l'approche habituelle consiste à projeter l'information colorimétrique sur le nuage de points ou sur la surface triangulée. Avec un nuage de points, cette approche ne permet un rendu visuel correct que si la densité locale du nuage de points est suffisante par rapport à la distance d'observation [Linsen, 2001]. Lorsque l'on dispose d'un modèle de surface, le rendu est très bon lorsque l'on observe la surface depuis une direction proche de la direction de prise de vue, mais devient mauvais lorsque l'on s'en écarte (fig. 3.5). Nous avons vu que l'affichage d'un modèle de surface non simplifié était complexe à gérer pour l'ordinateur. Il l'est encore plus lorsqu'il est texturé. De plus la résolution des photographies est en général supérieure à celle du nuage de points et les rééchantillonnages nécessaires à la projection de la photographie vont dégrader

cette résolution.

Le principe de l'image solide est de faire la démarche inverse. La photographie conserve sa géométrie initiale, non redressée de la distorsion, et les informations de position de chaque pixel sont ajoutées. Au lieu d'avoir seulement trois couches d'information correspondant aux composantes rouge, verte et bleue (et éventuellement d'autres informations radiométriques liés aux modes multispectraux additionnels), l'image possède des couches supplémentaires contenant l'information de position (fig. 3.6). L'information de position peut être stockée sous forme de trois couches X Y et Z, ce qui en rend l'interprétation instantanée. Elle peut aussi être stockée sous forme d'une seule couche de distance afin de minimiser la quantité de données à stocker. La distance stockée est alors la distance par rapport au centre de perspective photographique et le calcul de la position à partir de la distance se fait en utilisant les équations de photogrammétrie (1.3) à (1.5). Cette forme plus compacte nécessite donc de conserver les paramètres de position, d'orientation et de calibration relatifs à la prise de vue pour pouvoir être exploitée.

En plus des informations de couleur et de position, une couche supplémentaire peut être créée afin de stoker l'information de réflectivité issue des données laser. Cette information est rarement exploitable du fait d'un manque de contraste entre les différents matériaux, mais il peut arriver qu'une différence lithologique s'exprime par un contraste de réflectivité suffisant pour être exploité.

Lors de l'exploitation, l'utilisateur visualise la photographie comme dans un logiciel de dessin classique, ainsi que les coordonnées du pixel sur lequel il se trouve. Les couches d'information de position et de réflectivité ne sont pas visualisées en temps normal, l'utilisateur peut décider de les observer, mais ne voit plus dans ce cas l'information de couleur (fig. 3.9).

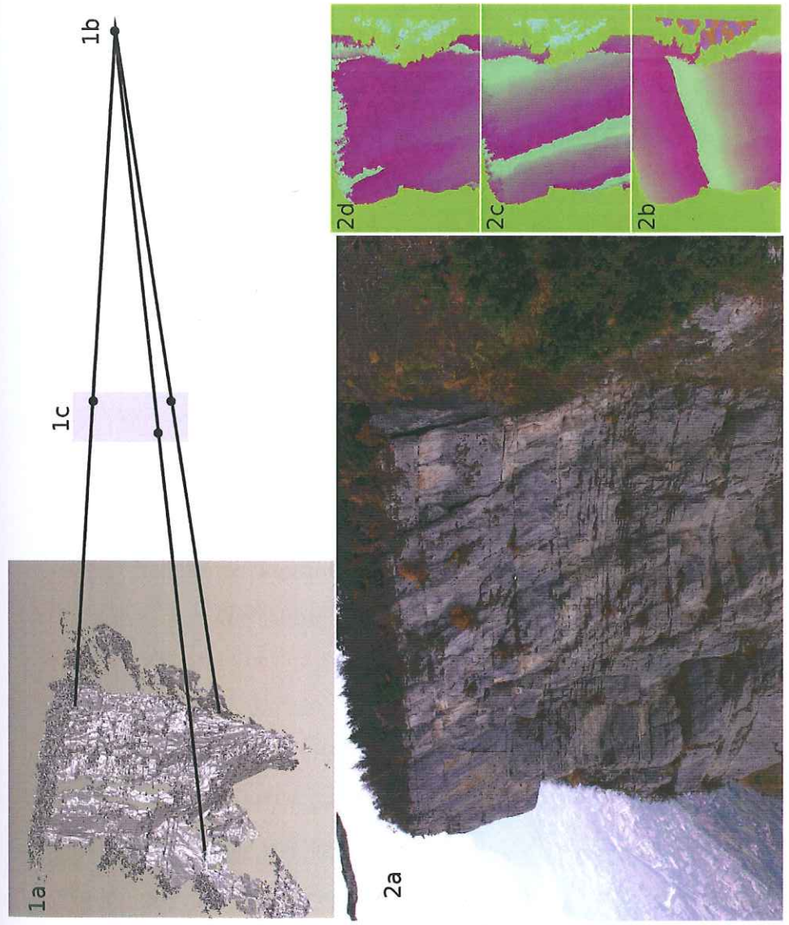


Fig. 3.6 – Concept de l'image solide.
1a : Objet dont on connait la position (ici la falaise du Ravin de l'Aiguille). 1b : Position de prise de vue photographique. 1C : photographie. On sait calculer la position de chaque pixel. 2 : Image solide créée. 2a : La photographie conserve sa géométrie initiale. La distorsion n'est pas redressée pour ne pas réechantillonner l'image. 2b,c,d En plus des couches d'information de couleur, trois couches d'informations supplémentaires donnent la position de chaque pixel.

3.1.2.2 Calcul de l'image solide

À cause de la distorsion optique et de la difficulté à faire coïncider exactement l'axe optique du laser et celui de l'appareil photographique, il n'est pas possible d'établir une correspondance directe entre les points obtenus par le laser et les positions des pixels. Ces positions doivent être calculées, soit par interpolation à partir du nuage de points, soit à partir du MNS.

3.1.2.2.1 Depuis un nuage de points. La création d'une image solide depuis un nuage de points permet de se passer de l'étape délicate de triangulation. Il n'est pas non plus nécessaire de supprimer la végétation du nuage de points, celle-ci étant généralement facilement identifiable sur la photographie.

Le principe de création de l'image solide est relativement simple : il faut d'abord calculer la position de chaque pixel de l'image. Pour ce faire il faut déterminer quels sont les points du nuage qui sont contenus dans chacun des pixels, puis interpoler la position du pixel en fonction des coordonnées de ces points.

La première étape se fait de la manière suivante : on parcours le nuage de points, et pour chaque point, on calcule sur quel pixel il se projette. Le calcul de la projection se fait en utilisant les équations (1.3) à (1.5) qui nous donnent des valeurs décimales pour X_p et Y_p. On affecte le point au pixel $\left(\bar{X}_p, \bar{Y}_p\right)$ le plus proche. Cette étape nous donne pour chaque pixel la liste des points qui s'y projette.

La deuxième étape consiste à interpoler la position du pixel. Si le nuage est suffisamment dense, chaque pixel comporte des points de part et d'autre du centre du pixel, et l'on peut se contenter d'interpoler la position des pixels à partir des points propres à chaque pixel.

Dans le cas contraire, que l'on rencontre le plus souvent, il faut aussi utiliser les points des huit pixels voisins. Un contrôle est cependant réalisé sur la radiométrie des pixels : les pixels dont la radiométrie est trop éloignée de celle du pixel dont on calcule la position sont considérés comme ne faisant pas partie du même élément et n'entrent pas en compte dans le calcul de l'interpolation. Cela permet, par exemple, à proximité d'une arête, de ne prendre en compte dans l'interpolation que les pixels qui sont du même coté de l'arête. Si il n'y a pas au moins quatre points dans les neufs pixels en question, aucune position n'est calculée pour ce pixel.

Lorsque les points de vue laser et photographiques sont identiques ou très proches, tous les points obtenus par le laser correspondent à des parties de la paroi visibles sur l'image. La méthode est alors théoriquement justifiée. Lorsque la photographie est prise depuis un point de vue différent du point de vue laser, il existe des cas de figure où la méthode est théoriquement infondée. Une situation de ce type a été rencontrée sur le Rocher du Midi. L'intérieur de la fissure principale a été partiellement imagé par le laser. Les points se trouvant dans la fissure sont projetés sur les pixels de la photographie. Comme la photographie est prise depuis un point de vue frontal, la fissure n'y est pas visible, se trouvant en arrière-plan de la paroi. Un point de la paroi A est projeté sur le même pixel qu'un point dans la fissure B lorsque B est situé sur la droite (OA) reliant le point de prise de vue et la paroi (fig. 3.7). Les points dans la fissure, sensés être invisibles depuis le point de vue photographique, interviennent donc à tort dans le calcul de la position des pixels de la paroi.

Plus généralement il suffit qu'une partie de la paroi soit visible depuis le point de vue laser et masquée depuis le point de vue photographique pour que le calcul de la position des pixels masquant soit erronée. Ceci arrive fréquemment dès que les points de vue sont éloignés l'un de l'autre, du fait de surplombs ou de rentrants visibles depuis un seul point de vue. Il faut alors trouver un moyen de discriminer entre les

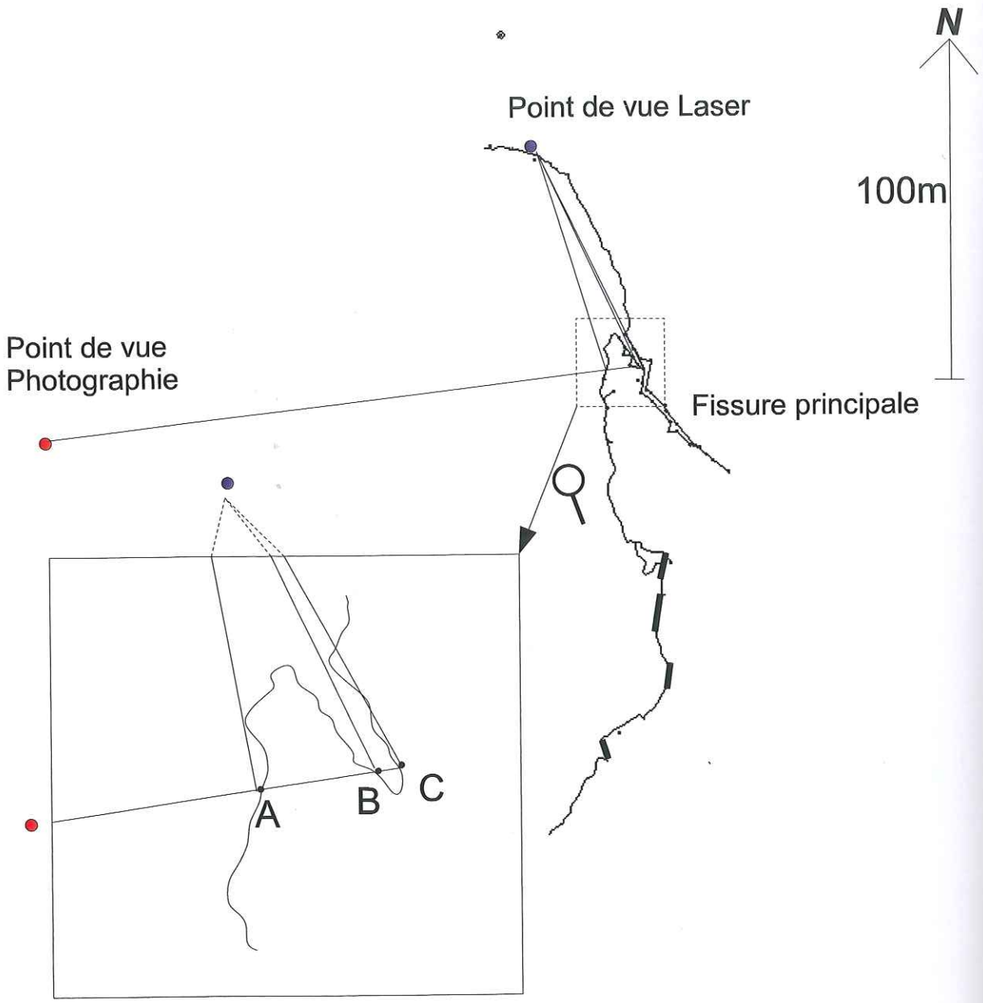


Fig. 3.7 – Difficulté de création de l'image solide lorsque les points de vue laser et photographique sont différents.
Le cas présenté provient du site du Rocher du Midi. Les points laser B et C, à l'intérieur de la fissure, correspondent à des parties de la paroi invisible depuis le point de prise de vue photographique. Pourtant ils interviennent dans le calcul de la position du pixel A

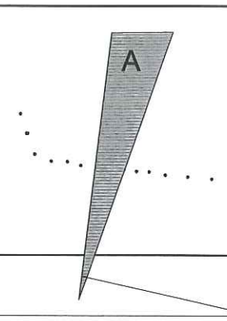
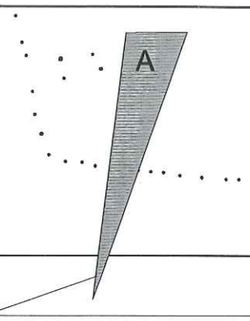
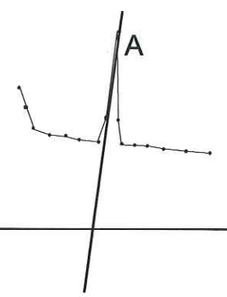
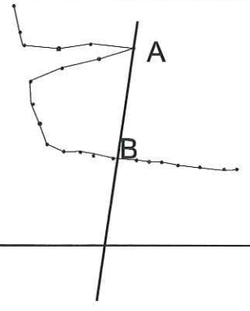
	Fissure dans l'axe de prise de vue.	Fissure dans une autre direction.
Depuis un nuage de points		
	Zones contenant les points à prendre en compte dans le calcul de la position du pixel P. Dans les deux cas le point A est loin en arrière des autres points.	
Depuis une surface triangulée		
	Demi droite se projetant sur le pixel P. Dans les deux cas la demi-droite rencontre un triangle tout près de A. Dans le cas de la fissure dans l'axe c'est le seul triangle rencontré. Si la fissure est dans une autre direction, un triangle est rencontré avant, en B, là où la demi-droite rencontre la paroi.	

Fig. 3.8 – Utilisation de la distance pour déterminer si un point est sur une partie cachée ou une partie visible de la paroi.
L'information n'est pas discriminante sur le nuage de points (a), mais le devient sur la surface triangulée(b).

points correspondant à un objet au premier plan et un objet en arrière-plan qui devrait être masqué depuis le point de vue photographique. La distance entre les points et la position de prise de vue, utilisée seule, n'est pas un bon discriminant car elle ne permet pas de faire la différence entre la situation avec un objet au premier plan et un objet au second plan et la situation avec un plan sub-parallèle à l'axe, entre le point proche et le point de prise de vue (fig. 3.8).

3.1.2.2.2 Depuis un modèle de surface. Pour résoudre ce problème, j'ai conçu un algorithme simple mais fonctionnel pour former une image solide en partant de la surface triangulée au lieu de partir du nuage de points. Les étapes de l'algorithme sont les suivantes :

1. Pour chaque triangle on détermine les pixels sur lesquels se projette le triangle.
2. Pour chacun de ces pixels on calcule la coordonnée de l'intersection entre la droite issue du centre de ce pixel et le triangle.
3. Si la distance entre ce point et le point de prise de vue est inférieure à la valeur en mémoire (initialisée à ∞) pour ce pixel, elle remplace la valeur en mémoire.
4. Une fois tous les triangles parcourus, on enregistre les valeurs de distance retenues pour tous les pixels. Si un pixel n'est à l'intérieur d'aucun triangle, aucune distance n'est calculé pour ce pixel.

Aucune interpolation n'est plus nécessaire puisqu'elle est déjà faite au moment de la triangulation. On considère que la surface des triangles correspond parfaitement avec la surface réelle de la falaise, ce qui revient à calculer la distance en réalisant une moyenne pondérée sur les trois points les plus proches.

L'étape permettant de palier le problème de différence de point de vue est la troisième étape de l'algorithme. En effet, si l'on se trouve au niveau d'un pixel ne correspondant pas à une partie cachée, un seul triangle intersecte ce pixel. Si le pixel se situe

sur une droite passant par une partie cachée, plusieurs triangles intersecteront le pixel. Dans le cas de la fissure du Rocher du Midi présenté sur la figure 3.8, trois triangles intersectent le même pixel : le triangle de la paroi, le triangle de la face coté falaise de la fissure et le triangle de la face coté massif de la fissure. Seul le triangle le plus proche du point de prise de vue est visible, les autres sont masqués par le premier.

Cette méthode permet de créer une image solide en partant d'une photographie depuis un point de vue distinct du point d'acquisition laser. Elle possède par contre deux inconvénients qui lui feront préférer l'autre méthode lorsque celle-ci est applicable. D'abord elle nécessite l'étape de triangulation dont nous avons vu qu'elle pouvait poser problème. Ceci n'est cependant pas forcément gênant, notamment dans le cas d'une triangulation à partir d'une unique acquisition fixe. Mais cela reste problématique si l'on compte utiliser plusieurs acquisitions différentes ou une acquisition faite avec un laser mobile, ce qui concerne en particulier les scannerisations héliportés. Le second inconvénient de cette méthode est lié au fait que l'interpolation a lieu au moment de la triangulation. Cette interpolation est donc effectuée sans exploiter l'information radiométrique contenue dans la photographie. À proximité des arêtes, l'interpolation est donc faite sur tous les points proches. Dans le cas d'un calcul directement depuis le nuage de points, cette interpolation est faite uniquement à partir des points de même couleur et est donc légèrement plus précise lorsque le problème des parties cachée ne se pose pas.

T2IS, le script développé afin de réalise le calcul de l'image solide depuis une surface triangulé, est joint en annexe.

3.1.2.3 Outils de mesures disponibles

3.1.2.3.1 Cadre logiciel. Le calcul de l'image solide à partir du nuage de points est réalisé dans le logiciel LSR développé au Politecnico de Turin. Ce logiciel possède une interface de visualisation de l'image solide ainsi qu'une possibilité de réaliser quelques mesures. Ayant des besoins spécifiques, et n'ayant pas la possibilité de développer nos propres application à l'intérieur du logiciel LSR, nous avons dû trouver un logiciel permettant de travailler sur des images multi-couches nous laissant la possibilité de développer nos propres outils de mesure. Il nous fallait aussi avoir la possibilité d'exploiter les images solides que nous avions nous-même crées à partir de surfaces triangulées. Ce calcul fait par un script perl, fournit le résultat dans un format ascii, alors que le format de donnée interne de LSR n'est pas exploitable.

Le logiciel sous licence libre ImageJ [Abràmoff *et al.*, 2004] nous est apparu comme un cadre de développement adéquat. C'est un programme conçu pour réaliser des calculs et des analyses statistiques sur les images. Le logiciel est écrit en Java et prévu pour que l'utilisateur puisse facilement développer de nouvelles fonctionnalités sous forme de greffons/plugins. Ce logiciel permet une portabilité immédiate (du logiciel mais aussi des greffons) sur tous les systèmes d'exploitation, ainsi qu'une simplicité dans la création de greffons. Cela rend par contre le logiciel moins rapide qu'un logiciel qui aurait été écrit dans un langage moins évolué comme le langage C.

Le choix a été fait de conserver l'information de position sous forme de trois couches X Y et Z plutôt qu'une seule couche de distance, la plus grande occupation de mémoire est compensée par l'absence de besoin de recalculer la position en permanence et une simplification lors d'éventuelles opérations de recadrage ou de mozaïquage d'image.

Les images multicouches sont gérées à condition que le format, en particulier le nombre de bits par pixels, de toutes les couches soit identiques. Si l'image initiale est

en couleur, l'information radiométrique est généralement enregistrée sur 24 bits, ce qui nous impose d'enregistrer l'information de position sur 24 bits par composante. En limitant la précision au mm, la plage de valeur utilisable pour chacune des couches X Y et Z est comprise entre -8000m et +8000m. Le logiciel affecte à un pixel pour lequel la position n'a pas pu être calculée la valeur (0,0,0).

Une image panchromatique est habituellement encodée sur 8bits, ce qui est bien insuffisant pour encoder l'information de position. On pourrait considérer l'image comme une image couleur et encoder la position sur 24bits. On peut aussi utiliser la capacité d'imageJ de traiter des images en niveau de gris 32bits pour coder notre information de position sur 32 bits, donc avec une précision et une plage de valeur plus élevée que pour les images en couleur.

Dans la pratique la précision millimétrique de l'enregistrement de la position pour les images couleur est supérieure à celle de la mesure laser et aucune image ne couvre une amplitude de 16km. Cependant, la plage de valeur limitée des images couleur ne permet pas de travailler directement en coordonnées géographiques complètes telles que celles utilisées dans le système Lambert par exemple. Il faut alors travailler dans un repère « local », soit avec l'origine prise à l'emplacement d'un des points de vue laser (cas du Rocher du Midi), soit une troncature des premiers chiffres des coordonnées Lambert (cas du Rocher de la Bourgeoise).

3.1.2.3.2 Affichage des coordonnées. Maintenant que nous avons décrit le cadre général dans lequel les greffons de mesure sur l'image solide sont développés, nous allons passer en revue les différents greffons qui ont été ajoutés au fur et à mesure de nos besoins. Les outils permettant de faire des sélections de différents secteurs de l'image sont déjà présents dans ImageJ (rectangle, ovale, polygone, polygone à main levée, droite, polyligne, ligne à main levée et point seul, avec la possibilité de faire des

sélections multiples) et la récupération des pixels contenu dans une sélection par un greffon est relativement simple.

Le greffon *ImageSolide* (cf. annexe 4) gère les interactions entre les autres greffons. Il permet entre autre de configurer des raccourcis claviers pour lancer chacun des autres greffons.

Le premier greffon développé, *MouseListenerb* (annexe 5), est un greffon qui permet d'afficher les coordonnées. Le logiciel récupère la position de la souris et affiche en continu les coordonnées du pixel sur lequel se trouve le curseur. Une bascule à chaque clic de souris permet de passer de l'affichage de la position du point à l'affichage de l'écart avec le dernier point cliqué. (fig. 3.9)

3.1.2.3.3 Calcul et représentation du plan moyen. Le greffon qui s'avère le plus utile pour les études structurales est celui qui permet le calcul du plan moyen, *planmoyen* (annexe 6). La base de ce greffon récupère l'ensemble des pixels de la sélection en cours et calcule le plan minimisant la distance quadratique moyenne entre le plan et les points.

Le plan a une équation de la forme $aX + bY + cZ + d = 0$ avec $\|a, b, c\| = 1$ et les n points des différents pixels ont des coordonnées $(X_i, Y_i, Z_i)_{i=1..n}$. Les valeurs a, b, c sont les cosinus directeurs de la normale au plan moyen. Par convention le logiciel affichera toujours la normale montante. La distance orientée entre le point P_i et le plan est $\delta_i = aX_i + bY_i + cZ_i + d$. On doit donc trouver les valeurs de a, b, c et d qui minimisent $\sum_{i=1}^{n} \delta_i^2$. La détermination de a, b, c, et d se fait de manière analytique à partir des paramètres suivant : $\sum X_i^2$, $\sum Y_i^2$, $\sum Z_i^2$, $\sum X_i$, $\sum Y_i$, $\sum Z_i$, $\sum X_i Y_i$, $\sum X_i Z_i$, $\sum Y_i Z_i$. Le greffon affiche l'équation du plan, le nombre de points utilisés (qui est différent du nombre de pixels sélectionnés si certains pixels n'ont pas pu être interpolés) et la

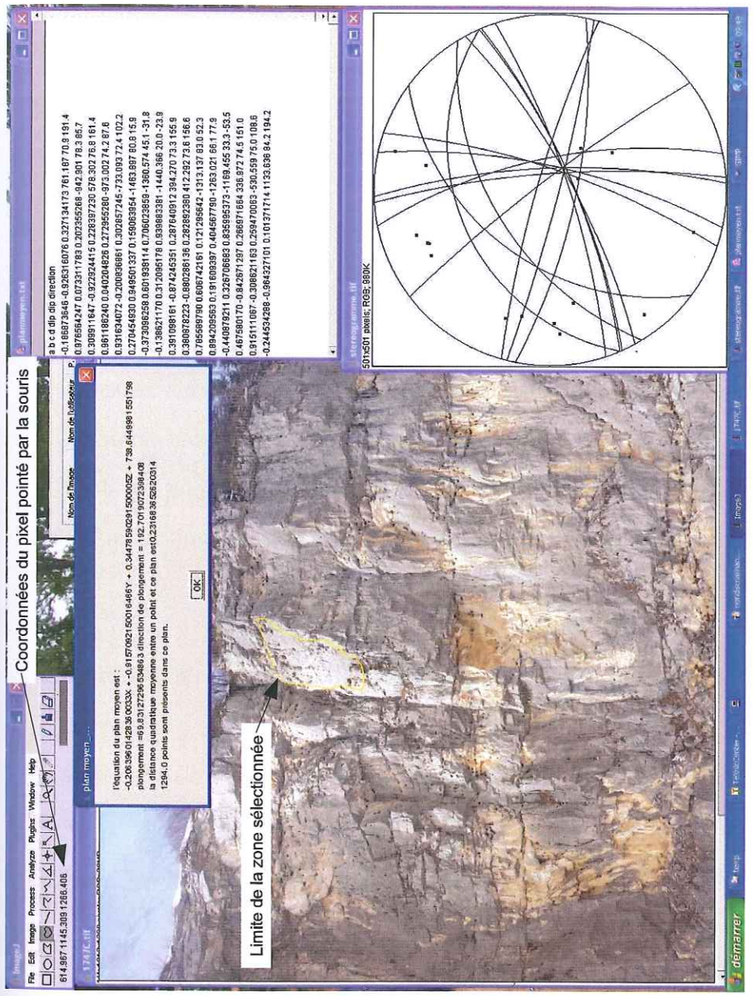


Fig. 3.9 – Utilisation des greffons de base de l'image solide dans le logiciel ImageJ. *Les coordonnées en haut à gauche, qui affichent ici la position du pixel pointé, peuvent être basculées en mode distance pour afficher les écarts $\Delta X\ \Delta Y\ \Delta Z$ et la distance avec un point précédemment sélectionné. Le greffon « planmoyen » calcule le plan correspondant à la zone sélectionnée (contour jaune). Le nombre de point présent est le nombre de pixel de la sélection ayant une position calculée. À l'issu du calcul l'équation du plan est ajoutée au fichier « planmoyen.txt » et sa trace (cycle et pôle) est ajoutée sur le fichier image* « stéréogramme.tif » .

distance quadratique moyenne $\sqrt{\frac{\sum_{i=1}^{n} \delta_i^2}{n}}$. Le plongement, qui vaut arccos c et la direction de plongement, qui vaut $\arctan a/b + 90 \times \left(1 - \frac{|b|}{b}\right)$ sont aussi affichés.

Lorsque la mesure est validée par l'opérateur, les paramètres a b c d de l'équation du plan ainsi que le plongement et la direction de plongement sont ajoutés sur une nouvelle ligne du fichier texte « planmoyen » . La trace du plan est ajoutée sur un stéréogramme (pôle et trace cyclographique) (fig. 3.9).

En cliquant sur une trace du stéréogramme, Le greffon *ImageSolide* rappelle sur l'image principale l'affichage de la sélection ayant servi à calculer ce plan, et affiche son équation (a b c d plongement direction de plongement). Pour pouvoir effectuer ce rappel d'affichage, chaque zone de sélection est préalablement enregistrée dans un fichier nommé d'après le numéro de la sélection, et chaque plan affiché sur le stéréogramme est dans une couleur légèrement différente. Le nombre de plans calculables est donc limité au nombre de couleurs, ce qui dans la pratique n'apporte pas de limitation.

3.1.2.3.4 Aire d'une sélection. Le greffon *surface* (annexe 7) calcule la surface d'une zone de sélection polygonale. Le greffon commence par calculer le plan moyen comme précédemment, puis il projette tous les points formant le périmètre de la sélection sur ce plan moyen, enfin il calcule l'aire du polygone formé comme la somme des aires des triangles formés par les sommets S_0,S_i et S_{i+1} (fig. 3.10). L'aire du triangle est calculée par la norme du produit vectoriel $A_{i-i+1} = \left\| S_i\overrightarrow{S_{i+1}} \wedge \overrightarrow{S_iS_0} \right\|$.

Si le polygone de sélection est convexe, l'aire globale est la somme des normes des produits vectoriels qui est aussi la norme de la somme des produits vectoriels : $A_p = \sum_i \left\| S_i\overrightarrow{S_{i+1}} \wedge \overrightarrow{S_iS_0} \right\| = \left\| \sum_i S_i\overrightarrow{S_{i+1}} \wedge \overrightarrow{S_iS_0} \right\|$. Si par contre le polygone n'est pas convexe, il peut arriver qu'une partie d'un triangle $S_0S_iS_{i+1}$ soit à l'extérieur du polygone. On a donc $A_p < \sum_i \left\| S_i\overrightarrow{S_{i+1}} \wedge \overrightarrow{S_iS_0} \right\|$ mais dans ce cas l'un des angles $S_0\widehat{S_iS_{i+1}}$ possède un

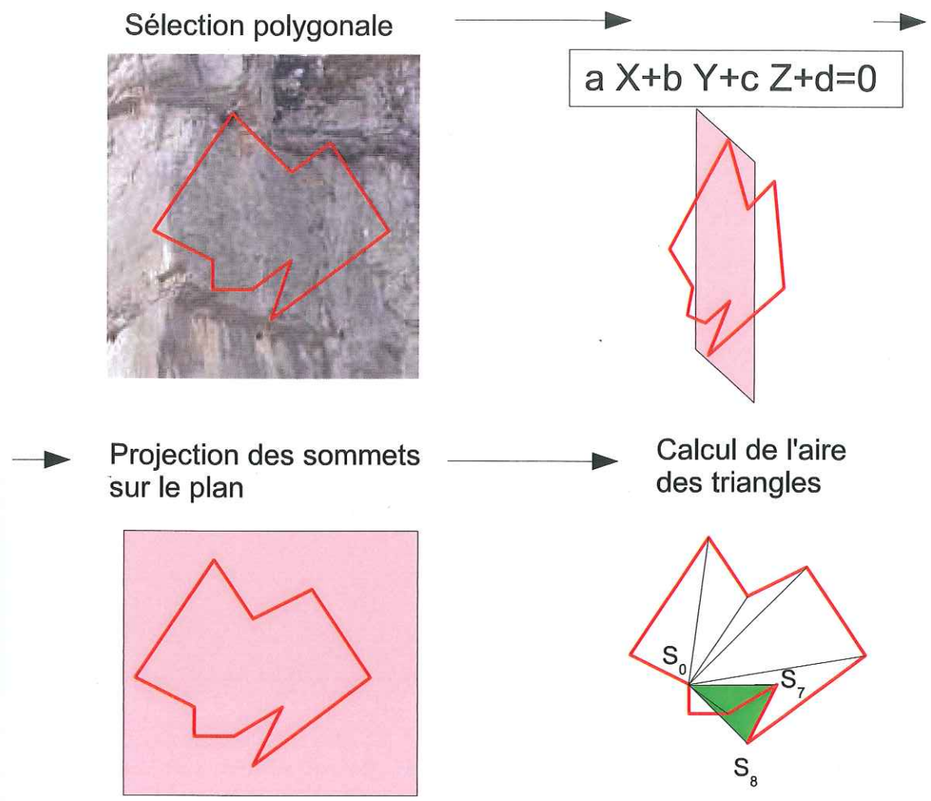


Fig. 3.10 – Calcul de la surface correspondant à une zone de sélection.
En premier lieu le plan moyen de la zone sélectionnée est calculé. Ensuite les sommets du polygone de sélection sont projetés sur le plan. Enfin les aires des triangles construits sur les cotés du polygone sont calculées et additionnées. L'aire du triangle en vert doit être soustraite au lieu d'être additionnée.

signe opposé aux autres, et le vecteur $S_i\overrightarrow{S_{i+1}} \wedge \overrightarrow{S_iS_0}$ est de sens opposé aux autres, et l'égalité $A_p = \left\| \sum_i S_i\overrightarrow{S_{i+1}} \wedge \overrightarrow{S_iS_0} \right\|$ reste vérifiée.

3.1.2.3.5 Enregistrement d'une sélection. Le greffon *listing* (annexe 8) permet d'exporter l'ensemble des pixels sélectionnés dans un fichier texte choisi par l'utilisateur. Les pixels non nuls sont enregistrés sur une ligne de texte au format : « point X,Y,Z ». Cet export est utilisé, entre autre, pour réccupérer dans d'autres logiciels le contour ou le tracé d'une courbe suivie sur l'image, par exemple pour récupérer la position du géoradar en suivant visuellement sur la photographie la descente effectuée. Il peut aussi être utilisé pour réaliser des coupes en exportant une sélection linéaire.

Le format d'export est fixé dans le fichier source qui définit le greffon, il n'est donc pas possible de le modifier à l'utilisation. Il est néanmoins simple de modifier le fichier source et de recompiler le greffon. Ceci est d'autant plus facile que la compilation des greffons est possible pendant l'utilisation d'ImageJ sans qu'il soit nécessaire de redémarrer le logiciel. Le format type illustré dans le greffon correspond aux commandes Autocad de définition d'un point, l'ensemble des points exportés depuis l'image solide est dans ce cas un fichier script Autocad

3.1.2.3.6 Intersection d'un objet avec la falaise. Il est parfois nécessaire de savoir à quel endroit un plan d'équation donné intersecte la falaise, soit parce qu'on a mesuré une fracture à un endroit de la falaise et que l'on souhaite savoir si son prolongement émerge à un autre endroit (fig. 3.11), soit parce que l'on a estimé la position et l'orientation d'une fracture à l'intérieur du massif par une technique d'imagerie géophysique et que l'on souhaite savoir à quel endroit cette fracture recoupe la surface de la falaise. On a aussi besoin de projeter sur l'image la position, connue par ailleurs, de certaines mesures ou observations. Deux greffons différents permettent de traiter ce

Fig. 3.11 – Intersection d'un plan de fracture avec la paroi.
La fracture est mesurée sur une zone bien nette à mi-hauteur (en noir). On visualise très bien la position où cette fracture risque de se prolonger sur toute la hauteur de la paroi (en blanc) si elle n'est pas interrompue. La végétation, en s'éloignant de la paroi, crée par endroit des artéfacts repérable sous la forme de points blancs épars.

genre de problèmes.

Le premier greffon, *trouveintersect* (annexe 9), prend en paramètre l'équation d'un plan P ainsi qu'une distance D. Il calcule pour chaque pixel de l'image solide la distance entre ce pixel et P. Si cette distance est inférieure à D, le pixel est affiché dans une couleur spécifique. L'image avec les pixels proche de P recolorée est affichée dans une nouvelle fenêtre pour ne pas modifier l'image de départ. Ce greffon peut être modifié sans trop de difficultés pour être adapté à tous les objets pouvant être définis par une équation ou un système d'équations relativement simples.

Le second greffon, *createline* (annexe 10),prend en paramètre une série de points considérés comme les segments successifs d'une polyligne, et les paramètres d'orientation interne et externe de la photographie. Il projette les segments de droite sur l'image en utilisant les équations de photogrammétrie (1.3) à (1.5), sans vérifier si le point projeté est effectivement proche du pixel sur lequel il se projette. Cela permet d'afficher à l'image la position d'un objet même si il est situé en avant ou en arrière de la paroi, et d'afficher rapidement des objets à géométrie complexe, mais l'intersection effective avec la paroi n'est pas testée.

3.2 Étude statistique des orientations de fractures

Nous nous intéressons ici à la mesure de l'orientation d'un grand nombre de fracture, pouvant être de petite dimension, et pour lesquels il n'y a besoin ni d'une très grande précision sur l'orientation ni de connaître la position

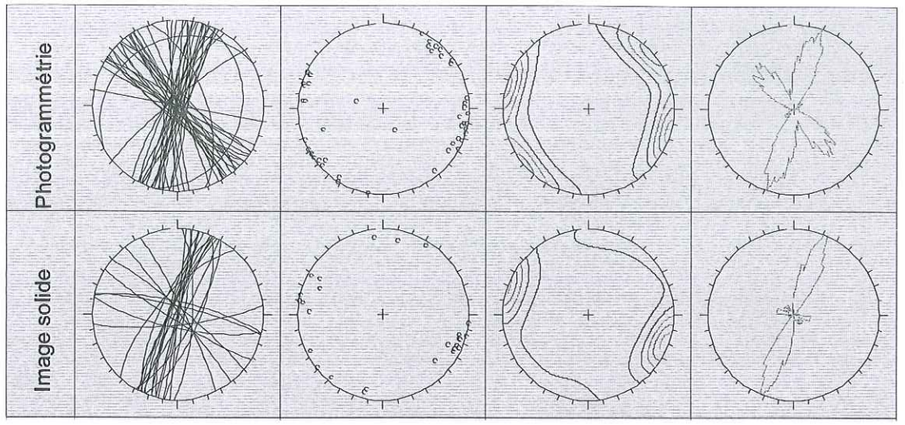


Fig. 3.12 – Relevé de fractures, Rocher du Midi.
De haut en bas, par mesure sur l'image solide et par photogrammétrie. De gauche à droite sont présentées : les traces cyclographiques des plans, les pôles des plans, les courbes d'isodensité de pôles et les diagrammes en rose.

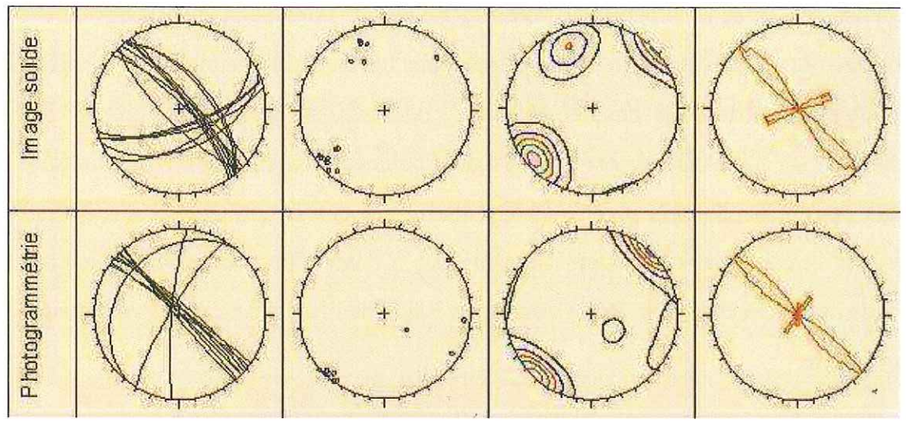


Fig. 3.13 – Relevé de fractures, Ravin de l'Aiguille.
De haut en bas, à la boussole, par mesure sur l'image solide et par photogrammétrie. De gauche à droite sont présentées : les traces cyclographiques des plans, les pôles des plans, les courbes d'isodensité de pôles et les diagrammes en rose.

Mesures : N ~~75~~-~~90~~° Fa
N130_140 Fb

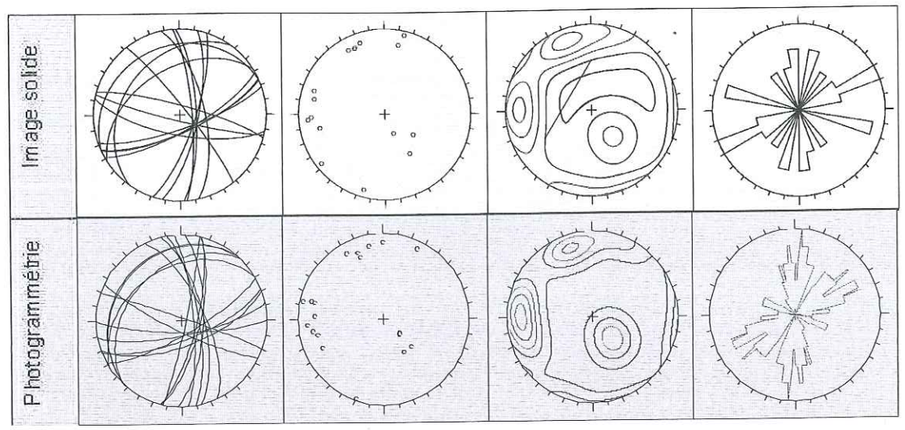


Fig. 3.14 – Relevé de fractures, Rocher de la Bourgeoise
De haut en bas, par mesure sur l'image solide et par photogrammétrie. De gauche à droite sont présentées : les traces cyclographiques des plans, les pôles des plans, les courbes d'isodensité de pôles et les diagrammes en rose.

So : N 40°E/45°W Fa : N140°E
Fb : N50°E

3.2.1 Approche photogrammétrique

Sur les trois sites du Rocher du Midi, du Ravin de l'Aiguille et du Rocher de la Bourgeoise, une analyse statistique des orientations des fractures visibles a été réalisée par photogrammétrie de la manière suivante : pour chaque plan de fracture repéré visuellement sur le modèlestéréoscopique, 4 ou 5 points ont été mesurés. L'orientation du plan passant le plus près possible de ces quatre points a ensuite été calculé. Cette analyse va être comparée au relevé classique fait à l'aide d'une boussole clinomètre pour les sites du Rocher de la Bourgeoise et du Ravin de l'Aiguille et au relevé fait sur l'image solide.

Sur le site du Rocher du Midi (fig. 3.12), on identifie trois familles de fractures. Les joints de stratification, subhorizontaux avec un léger pendage amont (figure 3.12), sont délicats à mesurer. Ils forment des surplombs dont il est difficile de visualiser le fond afin de réaliser une mesure précise. Celle-ci est cependant possible en deux endroits de la zone d'étude. L'orientation trouvée est proche de celle mesurée à la boussole sur le

plateau, mais le faible nombre d'individus mesurés tant par photogrammétrie que par les méthodes classiques empêche de réaliser une comparaison significative. Des prises de vue de la paroi en contre plongée, soit depuis un hélicoptère volant plus bas que pour les photographies exploitées, soit depuis le sol, auraient permis de mieux mettre en évidence et de mesurer les surplombs et l'orientation de la stratigraphie.

Les deux autres familles de discontinuités *Fa* et *Fb*, visibles sur la figure 3.12, sont des familles de fracture sub-verticales avec des horizontales orientées respectivement N10° à N30° et N110° à N140 °. L'orientation trouvée est, en moyenne, identique à celle trouvée par le relevé classique (fig. 2.4). On observe cependant un peu moins de variabilité sur le pendage des fractures que par le relevé classique sur le plateau, lié au plus petit nombre d'individus mesurés.

Sur le site du Ravin de l'Aiguille (fig. 3.13), on retrouve également trois familles de fractures. La stratification (fig. 3.13) n'est mesurée qu'en un endroit avec une orientation N308°E 26° compatible avec la mesure faite à la boussole. Bien qu'exprimée uniquement sous forme de surplomb, elle a pu être mesurée car des surplombs de grande dimension sont présents en haut de la falaise, au dessus de la hauteur de vol de l'hélicoptère. Deux autres familles *Fa* et *Fb* , déjà décrites au paragraphe 2.2.2.1, sont présentes.

↑N
Fa
Fb

La famille *Fb* est mesurée sans problème et l'orientation trouvée ne diffère que de 5° en moyenne de celle trouvée par le relevé classique. Les fractures de la famille *Fa* apparaissent difficilement sur les modèles stéréoscopiques : en effet, surtout présentes sur la moitié W du dièdre, elles sont alors masquées, dans un effet de perspective, par les indentations de la paroi qu'elles contribuent à former. Les trois fractures de cette famille qui ont pu être mesurées sont les trois individus présentant le plus de déviation vers le nord par rapport à l'orientation moyenne de cette famille. Les trois valeurs sont

compatible avec la variabilité observée pour cette famille par les autres méthodes, mais elles ne sont pas représentatives de l'orientation moyenne de cette famille. En effet les traces cyclographiques mesurées pour la famille *Fa* par photogrammétrie sont situées en bordure du faisceau de traces mesurées pour cette même famille par image solide et par boussole clinomètre (fig. 3.13).

Sur le site du Rocher de la Bourgeoise, la stratification (fig. 3.14) ainsi que les deux autres familles de fractures décrites au paragraphe 2.3.1 sont mesurées sans difficulté particulière et avec des orientations similaires à celles mesurées par image solide.

La mesure par photogrammétrie est suffisamment précise pour permettre de faire un relevé de fracturation, cependant les mesures sont lentes et fatigantes, surtout pour un opérateur qui n'est pas habitué au pointé stéréoscopique.

Selon la configuration du site et la position d'où sont prises les photographies, il est possible que des familles de fractures soient masquées. Pour empêcher ce phénomène il faut disposer de points de vue suffisamment nombreux et différents pour qu'aucune partie ne reste masquée, permettant des prises de vue en plongée et en contre plongée, depuis la droite et la gauche de l'objet étudié.

3.2.2 Image solide

Comme dans l'approche photogrammétrique, la fracturation a été étudiée par image solide sur les trois sites du Rocher du Midi, du Ravin de l'Aiguille et du Rocher de la Bourgeoise.

Sur le site du Rocher du Midi, la mesure des fractures a été réalisée uniquement sur la zone d'étude encadrée en rouge sur la figure 2.2, donc presque uniquement sur

la zone éclairée par le point de vue nord. La photographie utilisée comme support est la photographie aérienne n°24. La famille *Fa* (en rouge sur la figure 3.12) est détectée sans problème notable.

La résolution de l'image a dû être réduite pour qu'une position puisse être calculée pour un nombre raisonnable de pixels, et les rares surplombs qui avaient pu être mesurés par photogrammétrie ne sont plus assez grand pour être détectés par cette technique. Par ailleurs, la paroi est trop plane sur la zone étudiée pour que le suivi d'une ligne d'intersection entre le plan de stratification et la falaise permette de déterminer l'orientation de ce plan avec précision.

Des plans appartenant à la famille *Fb* sont identifiables aisément sur la photographie, mais ils correspondent à des plans masqués depuis le point de vue laser nord, sur lesquels la position des pixels n'a pas pu être calculée et donc pour lesquels le calcul du plan moyen n'est pas possible. Alors que la famille *Fb* est aisément identifiable sur les données issues de la photogrammétrie et du relevé classique, cette famille ne l'est pas sur les données issues de l'image solide. Quelques mesures (en bleu sur la figure 3.12) on pu être faites, mais elles sont trop peu nombreuses et trop dispersées pour identifier clairement une famille distincte de la famille *Fa*. Sur les courbes d'isodensité de pôle issu de l'analyse de l'image solide, on visualise une zone de très forte densité pour la famille *Fa* ainsi qu'une zone de densité faible mais non nulle englobant la famille *Fa* et la position théorique de la famille *Fb*. Sur les courbes d'isodensité correspondant aux autres méthodes, la zone de faible densité est identique, mais deux pôles de forte densité bien distincts sont visibles et correspondent aux familles *Fa* et *Fb*.

L'usage d'une photographie prise depuis un meilleur point de vue que le point d'acquisition laser, telle que la photographie aérienne, n'est donc que rarement productif puisque seuls les plans bien visibles depuis le point d'acquisition laser peuvent être me-

surés. Cet usage n'est productif que si plusieurs acquisitions lasers ont permis d'obtenir une densité de points suffisante sur toute paroi.

Sur le site du Ravin de l'Aiguille (fig. 3.13), l'analyse de fracturation a été réalisée sur deux séries d'images solides. Chaque série correspondant à l'un des points de vue d'acquisition laser (fig. 2.12). Le pied du dièdre n'est pas visible sur la même photographie que le sommet, donc pour chaque série, deux à trois images sont nécessaires pour couvrir toute la hauteur du dièdre, même si la majorité des mesures a été effectuée en haut de la paroi, dans les bancs de calcaire massif où les plans de fracture sont plus facilement identifiables.

La famille *Fa* (en rouge sur la figure 3.13) n'est pas ou mal visible depuis le point de vue 2 car ces plans sont parallèles à l'axe de prise de vue. Elle est par contre aisément mesurée depuis le point de vue 1. La famille *Fb* (en bleu sur la figure 3.13) est par contre mesurable depuis les deux points de vue. Les plans de stratification, en surplomb, ne sont visibles depuis aucun des deux points de vue, mais leur orientation peut être déterminée en suivant la trace d'une base de banc sur une grande longueur de paroi (fig. 3.15). Le relief de la paroi fait que la courbe obtenue n'est pas une droite. L'orientation de la stratification ainsi déterminée est de N307°E 24°. Cette valeur est très proche de la moyenne des valeurs trouvées par les deux autres méthodes. La possibilité de mesurer une orientation de plan par la sélection d'une ligne doit cependant être utilisé avec précaution car l'orientation du plan de mesure peut être biaisée dans les conditions que nous verrons au paragraphe 3.4.2.

Sur le site du Rocher de la Bourgeoise, les trois familles (*S0*, *Fa*, *Fb*) ont pu être mesurées et les orientations trouvées sont cohérentes avec les autres méthodes (fig. 3.14). Cependant, ces résultats nécessitent d'être détaillés. De part la moindre résolution de la scannerisation héliporté, la position des pixels n'a pu être correctement extrapolée

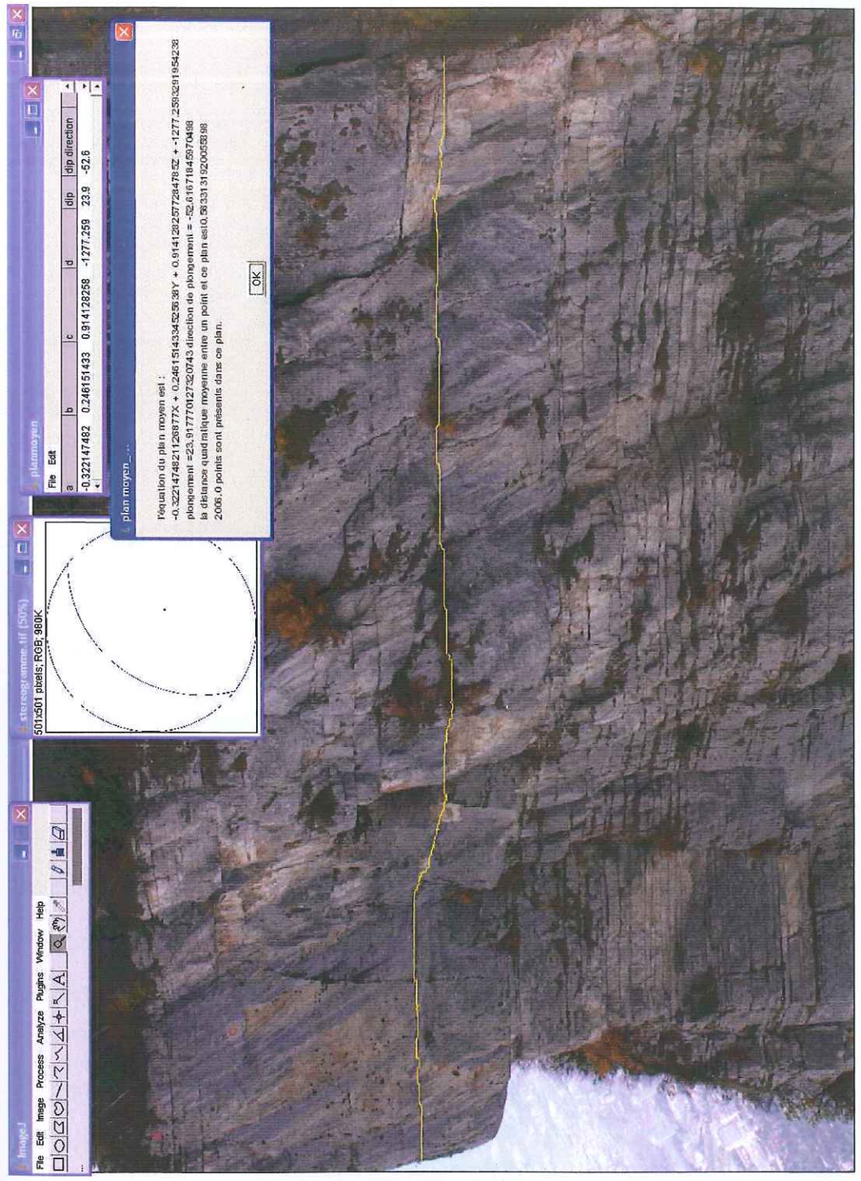


Fig. 3.15 – Mesure de l'orientation de la stratification à partir d'un joint de stratification. *Site du Ravin de l'Aiguille. La courbure de la paroi permet d'obtenir l'orientation du plan.*

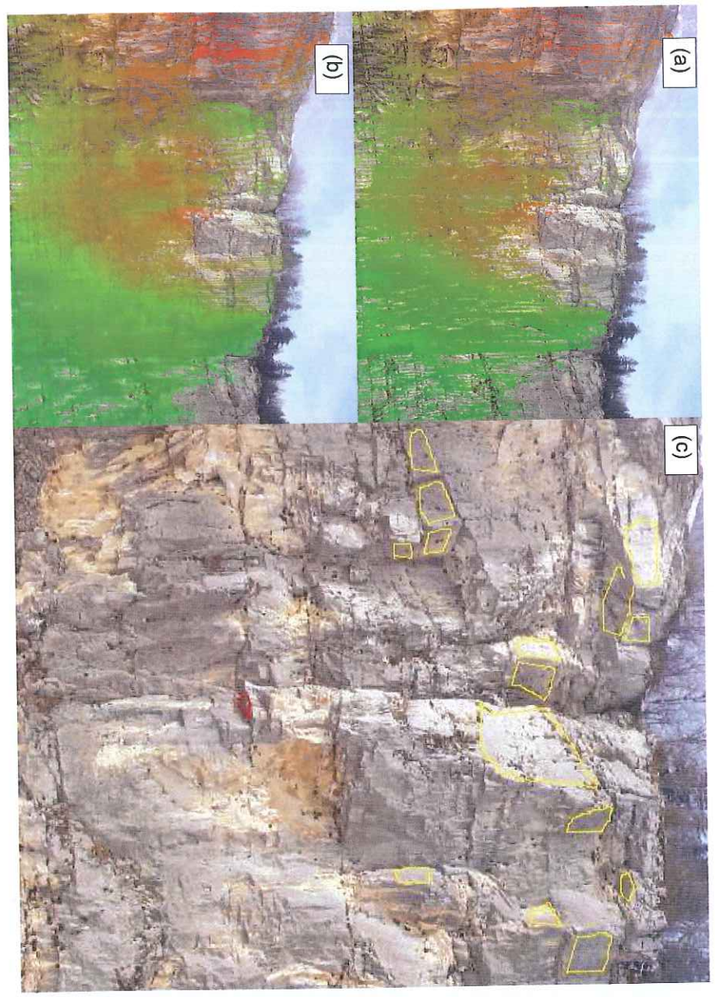


Fig. 3.16 – Illustration du manque de résolution de la scannerisation héliporté par rapport aux photographies sur le site du Rocher de la Bourgeoise
(a) et (b), superposition de la couche de distance sur les images solides du Rocher de la Bourgeoise. Les pixels pour lesquels une position a pu être calculée sont colorés en dégradés de rouge et vert, ceux pour lesquels aucune position n'est calculée sont visibles dans leur couleur initiale. Lorsque la résolution de l'image est de 2720×2040 *pixels (image a), 60% des pixels sont encore sans position. Pour* 1813×1360 *pixels (image b), la position est calculée pour tous les pixels des zones couvertes par les quatre passages Mais la densité de pixels calculés reste faible dans les zones où seul un à deux passages ont été réalisé. Le taux de pixels calculé est de 66%*
(c) Les plans mesurés pour l'étude de la fracturation sont entourés en jaune. le plus petit, en rouge, mesure $6,4m^2$

que pour des images ayant une résolution trois fois inférieure à la résolution initiale des images. Même ainsi, le nombre de pixels dont la position est connue reste faible dans les zones couvertes seulement par un des quatre passages hélicoptères (fig. 3.16). Les plans mesurés (contours en jaune sur la figure) sont des plans de grande dimension puisque le plus petit mesure $6m^2$.

Sur ce site, les trois familles de discontinuités présentent des grands plans permettant des mesures suffisamment nombreuses, mais l'étude par image solide ne nous permettrait pas de mesurer une famille de discontinuités présente sous la forme de faces de petite dimension, alors que la photogrammétrie réalisée avec les photographies en pleine résolution le permettrait.

3.2.3 Mesure directe sur le nuage de points ou le MNS

Les mesures d'orientation de discontinuités peuvent aussi être réalisées sur le nuage de points ou sur le modèle de surface. La figure 3.17 montre les limites de la méthode. Le plan est facile à mesurer une fois identifié comme tel, mais l'identification d'une zone du nuage de points ou du modèle de surface comme étant un plan de fracture n'est possible que si ce plan est de grande dimension par rapport à la distance entre les points et à l'incertitude sur la position des points. Si le nuage de points ou le modèle de surface n'est pas texturé à partir d'une photographie, l'identification des plans de fracture y est très délicate. Si le modèle est texturé, l'identification devient plus facile. La résolution de la photographie étant supérieure à celle de la scannerisation laser, des plans de trop petite taille pour être identifiés sur le nuage de point ou le modèle de surface pourront l'être sur l'image solide ou le couple stéréoscopique.

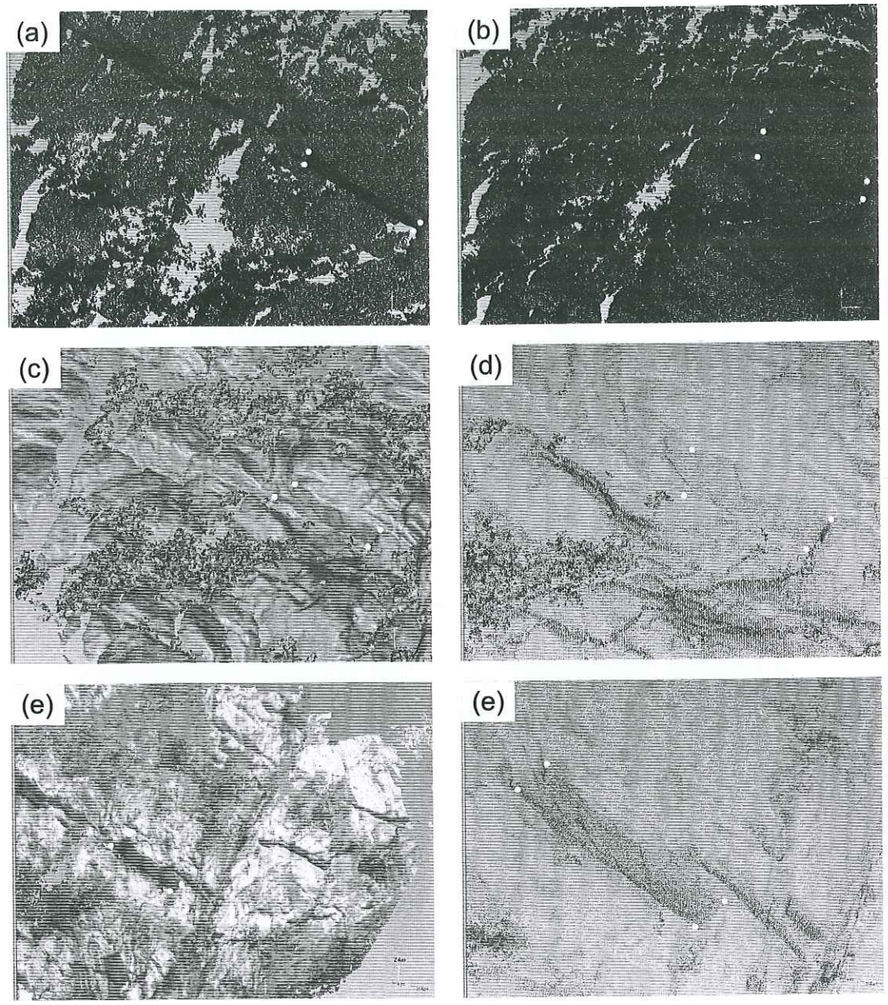


Fig. 3.17 – Mesure d'un plan de fracture de grande dimension (5000 points) sur le nuage de points, sur le site de Paganin.
Les quatres extrémités du plans sont marquées par un rond blanc. (a) et (b), nuage de point non colorisé. Vu de manière rasante(a), le plan est identifiable à une plus grande densité de point. Si le point de vue change légèrement (b), le plan disparaît. (c) et (d) nuage de point ombré en fonction de l'orientation locale de la paroi. Si le nuage est suffisamment dense(c), le plan est identifiable. Si l'on souhaite regarder le plan de plus près afin d'en préciser le contour (d), les points se séparent et le plan n'est plus identifiable. (e) et (f), nuage colorisé d'après une photographie. Les contours du plan sont un peu plus nets qu'en (c) et (d), mais la limite du plan reste malaisée à définir.

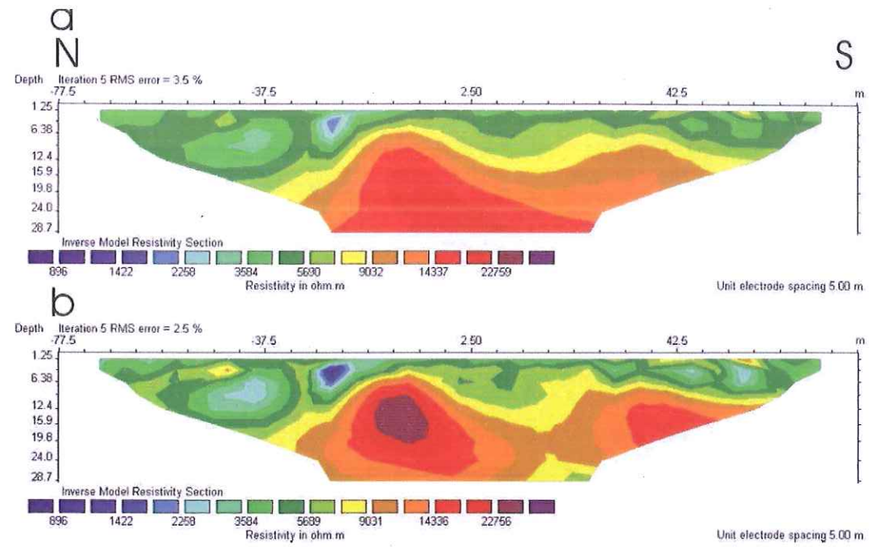


Fig. 3.18 – Apport de la prise en compte de la falaise dans les données de tomographie électrique du Ravin de l'Aiguille.
La tomographie présentée est celle du profil PE5 du Ravin de l'Aiguille (cf. fig. 2.13). La tomographie (a) est réalisée sans prise en compte de la falaise, (b) avec. La forme qualitative est similaire, mais les contrastes de résistivité sont nettement plus important en (b). La falaise est à plus de 50m de distance au point le plus proche [Jongmans et al., *2007].*

3.3 Apport des méthodes de télédétection rapprochée en support de la prospection géophysique

3.3.1 Imagerie sismique et électrique par tomographie

Les études géophysiques réalisées sur le replat en sommet de paroi sont sensibles à la position de la paroi. En sismique par exemple, on observera une réflexion sur l'interface entre la paroi et le vide. Comme la position de la paroi est connue avec précision, les données peuvent être corrigées de l'effet de l'interface formée par la paroi. De cette manière, les informations recueillies sur l'état de l'intérieur du massif sont

plus précises.

Sur le site du Ravin de l'Aiguille, des tomographies électriques ont été réalisées sur le plateau. La présence d'un vide, non conducteur électriquement, situé latéralement par rapport à la ligne de tomographie a une influence sur les résistivités locales mesurées. La distance à la paroi n'est pas constante sur toute la longueur du profil électrique, si bien que l'influence du vide est plus ou moins grande le long du profil. Une correction a été apportée sur les données des tomographies afin de tenir compte de la proximité de la paroi [Sahbi *et al.*, 1997]. La résolution des tomographies étant nettement inférieure à celle de la scannerisation laser de la paroi, un modèle simplifié de celle-ci est suffisant pour les corrections.

En l'occurrence la paroi étant sub-verticale, et l'influence de la paroi ayant lieu dès que celle-ci est rencontrée (donc dès le sommet), la correction a été faite en considérant une paroi parfaitement verticale. Une coupe horizontale a été réalisée en tête de paroi sur l'image solide, les corrections ont été faite sur la base de cette coupe en considérant la paroi totalement verticale. La figure 3.18 montre la différence sur le résultat de l'inversion des données selon que l'on prend en compte ou non l'influence de la paroi. Bien que d'aspect général proche, des différences significatives existent entre les deux tomographies. Sur les quatre profils plus proches de la paroi, son influence est encore plus grande et l'intérêt d'une correction est renforcé.

3.3.2 Géoradar en paroi

Le géoradar est, dans son principe, proche de la méthode de sismique réflexion. Un transmetteur (Tx) émet des ondes radar qui pénètrent dans le sol et se réfléchissent, et le signal résultant est enregistré par un récepteur (Rx). Par contraste avec les méthodes sismiques, les instruments radar utilisent des ondes électromagnétiques (EM) dont la

fréquence est beaucoup plus élevée et varie entre 25 MHz et 2 GHz. La pénétration bien plus faible qu'en sismique (seulement quelques dizaines de mètres pour du calcaire fracturé) est compensée par une bien meilleure résolution (25cm à 100 MHz). Cette technique est limitée lorsqu'elle est appliquée sur le replat en sommet de paroi parce que l'on ne détecte que les fractures sub-horizontales et que l'on est gêné par la couche altérée conductrice, à la surface du plateau. Cette technique ne donne donc ses meilleurs résultats que lorsqu'elle est appliquée directement sur la paroi [Deparis, 2007].

Une des difficultés qui doit donc être résolue pour exploiter les données du géoradar est la détermination de la position des antennes à chaque instant de l'acquisition. L'important masque généré par la paroi empêche de positionner les antennes par GPS. Les antennes étant descendues dans la paroi le long d'une corde de rappel, la distance séparant les antennes du sommet est connue, permettant de déterminer la position des antennes si l'on suppose que la falaise est régulière et que la corde de rappel suit bien la ligne de plus grande pente.

Pour déterminer plus précisément la position du géoradar, deux protocoles différents ont été utilisés. Le protocole le plus simple a été utilisé sur le Rocher de la Bourgeoise, où les acquisition géoradar avaient été faites indépendamment. Le tracé suivi par le géoradar a été sélectionné sur une image solide par l'opérateur ayant manipulé le radar dans la paroi et la trace correspondantes a été exportée. On connaît ainsi la courbe $(X, Y) = f(Z)$ suivie par l'opérateur au cours de la descente (fig. 2.17(b)). Comme on connaît aussi la distance au sommet à chaque instant, $\sqrt{(X - X_0)^2 + (Y - Y_0)^2 + (Z - Z_0)^2} = f(t)$, mesurée par la longueur de câble déroulé jusqu'à l'antenne, il est facile de calculer la position du géoradar à chaque instant $(X, Y, Z) = f(t)$.

Un deuxième protocole a été utilisé sur le Rocher du Midi. Des cibles réfléchissantes ont été disposées régulièrement le long de la descente en rappel du géoradar. Grâce à

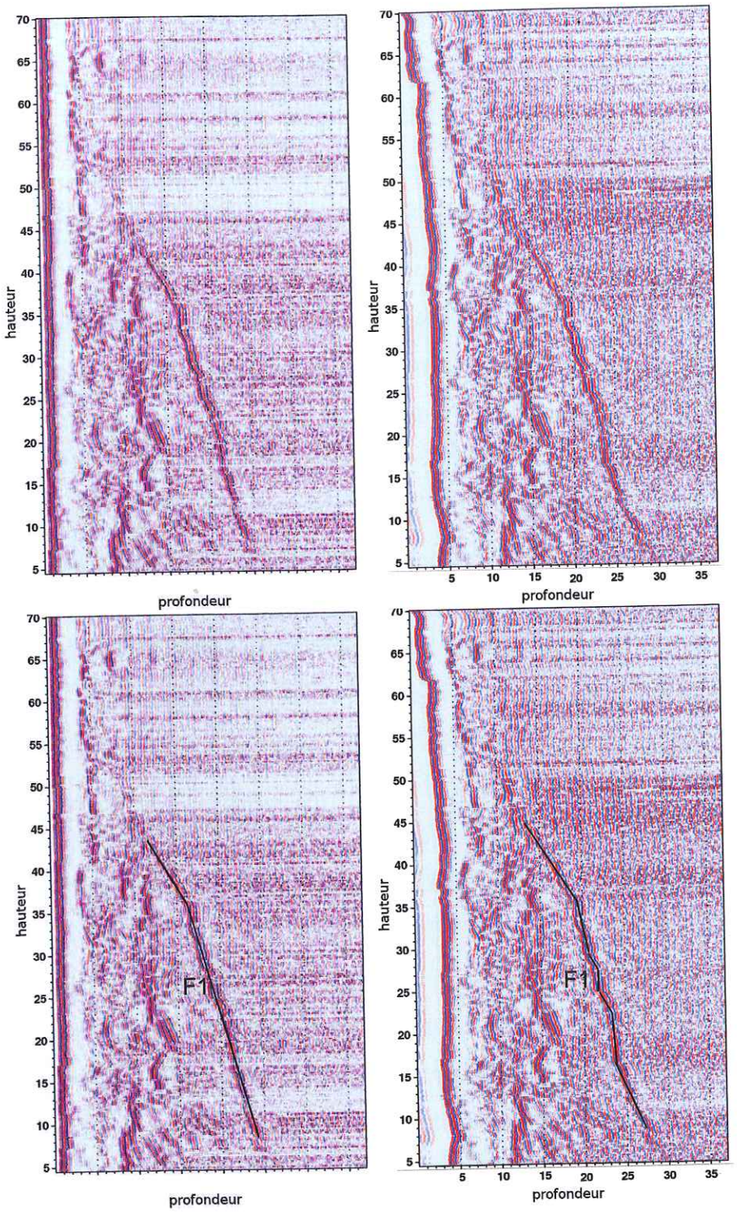


Fig. 3.19 – Profil géoradar P1 du Rocher du Midi réalisé à 100 MHz.
À gauche la correction du relief de la paroi n'est pas effectuée, à droite la correction est effectuée. le réflecteur principal, noté F_1 n'appartient pas aux familles de fractures visible à l'affleurement. Presque rectiligne sur le profil non corrigé (gauche), il présente des ruptures de pente plus marqué sur le profil corrigé pouvant correspondre à des relais entre deux plans de fractures différents. d'après Deparis, 2007.

leur forte réflectivité, Ces cibles sont repérées dans le nuage de points. Il faut ensuite déterminer la position prise par le géoradar entre ces points. Cela peut être fait par projection du segment reliant les deux cibles sur la paroi, la direction de projection étant la perpendiculaire locale à la paroi. Si les points sont suffisamment proches, la perte de précision par rapport au trajet le plus court entre ces deux points est négligeable devant la précision de la mesure laser d'une part et les variations latérales de la position d'autre part. La plupart des logiciels commerciaux de modélisation de surface (Géomagic, Rapidform, 3dReshaper, etc) proposent des outils de projection d'un segment sur la surface dont le résultat est d'une précision amplement suffisante pour les besoins de positionnement du géoradar.

L'intérêt de ce positionnement précis est double, tout d'abord les discontinuités imagées par le radar sont placées plus précisément, mais surtout leur orientation apparente diffère légèrement. La figure 3.19 montre le même profil radar avec ou sans prise en compte du relief de la paroi. Le réflecteur principal (noté F_1) présente pratiquement le même pendage moyen, mais sur des sections de quelques mètres la différence peut être importante. Ainsi sur la partie basse entre 8 et 15m, le pendage apparent mesurée et de 73° sur le profil brut et de seulement 65° sur le profil corrigé. Plus significatif encore, alors qu'il semble continu sur le profil sans correction du relief, il montre une alternance de parties plus raides et moins raides sur le profil corrigé. Cette alternance permet de supposer que ce réflecteur est composé de deux plans de fractures travaillant en relais, et non d'un seul grand plan d'orientation continu.

3.4 Cartographie des discontinuités majeures

Pour l'étude de la stabilité de la paroi (paragraphe 3.5), la mesure des paramètres statistiques de fracturation décrite en 3.2 est accompagnée par la mesure de l'orien-

tation et de la position des discontinuités majeures. Cette mesure peut être faite par photogrammétrie, par image solide ou directement sur le nuage de points. Ces trois méthodes donnent des résultats très proches comme le montre le tableau 3.1

3.4.1 Photogrammétrie

La mesure de la position et de l'orientation d'un plan de fracture majeur peut être effectué de différentes manières. La plus rapide consiste simplement à mesurer la position de trois points de la fracture et à calculer le plan passant par ces trois points. Plus les points sont loin les uns des autres, plus la mesure est précise. Cette méthode ne nous permet cependant pas d'apprécier si la fracture s'écarte beaucoup du plan modélisé (pas de mesure d'écart entre les points et un plan). Elle est de plus très sensible aux erreurs de mesure.

Pour que la mesure d'un point s'écartant de la surface moyenne de la fracture ne risque pas de fausser le résultat, on peut mesurer un grand nombre de points au lieu des trois nécessaires au calcul d'un plan. Les mesures seront alors réparties autant que possible sur toute la surface visible de la discontinuité. Le plan considéré au final sera le plan moyen défini de la même manière qu'en 3.1.2.3.3. La mesure manuelle de la position d'un point étant relativement longue, on utilise en général des méthodes semi-automatiques : asservissement au sol par corrélation numérique, corrélation automatique le long d'un profil mesuré manuellement, etc, pour densifier le nombre de mesures. Même ainsi, la mesure est considérablement plus longue que la simple mesure d'un plan par le calcul de la position de trois points. En se limitant à environ cinquante points par plan, il m'a fallu quatre fois plus longtemps que pour la mesure complètement manuelle sur trois points et dix fois plus que pour la mesure sur image solide.

	Photogrammétrie 3 points	Photogrammétrie 50 points	Image solide	Mesure sur le modèle de surface
Équation plan 1	$0.343x - 0.850y + 0.400z - 442.936 = 0$	$0.372x - 0.865y + 0.336z - 352.941 = 0$	$0.369x - 0.846y + 0.386z - 421.079 = 0$	$0.476x - 0.803y + 0.359z - 398.549 = 0$
Équation plan 2	$+0.670x + 0.718y - 0.190z - 236.755 = 0$	$0.832x + 0.504y + 0.231z - 346.652 = 0$	$0.732x + 0.674y + 0.103z - 173.499 = 0$	$0.663x + 0.744y + 0.082z - 145.358 = 0$
Équation plan 3			$0.569x + 0.690y + 0.447z - 635.926 = 0$	$0.582x + 0.703y + 0.408z - 585.052 = 0$
Temps de mesure total	10 minutes (2 plans)	40 minutes (2 plans)	4 minutes	6 minutes

Tab. 3.1 – Résultat des mesures des discontinuités majeures sur le site du Ravin de l'Aiguille.
Les résultats sont très proches quelle que soit la méthode utilisé. La plus grande différence d'orientation est de 5°. Si les résultats sont proches, les temps de mesure sont eux très différents, la photogrammétrie étant bein plus lente que les deux autres méthodes, même avec un nombre de points réduits

3.4.2 Image solide

La mesure de l'orientation d'un plan se faisant toujours par le calcul de l'orientation et de la position du plan moyen de la zone sélectionnée, la mesure de la position précise d'une discontinuité majeure ne diffère pas de la mesure d'un plan à des fins de mesure statistique d'orientation. Lorsque le plan de fracture est bien visible sur la photographie, la mesure est fiable et rapide. Si le plan est très rasant par rapport à la photographie, il devient difficile de sélectionner la totalité du plan sans sélectionner de pixels extérieurs au plan. Si la sélection englobe des points de part et d'autre du plan, l'orientation du plan calculé sera systématiquement différente de celle du plan réel. Par exemple, pour un surplomb ou l'on sélectionnerait des plans au-dessus et en-dessous du surplomb, le plan calculé sera plus vertical que le plan réel (fig. 3.20). Un plan vu selon un angle d'incidence nul ne sera pas directement mesurable.

Comme pour l'étude de fracturation, il faut donc veiller au moment de l'acquisition laser à disposer de suffisamment de points de vue différents pour que les principales discontinuités soient toutes acquises depuis un point de vue non rasant. Si ces discontinuités apparaissent à l'affleurement sous forme de plan de grande dimension, l'angle d'incidence minimum peut être plus faible que celui requis pour l'étude statistique des plans. Sur le site du Rocher du Midi, où la station laser se trouve à 100m de la zone d'étude, un plan vu sous une incidence presque rasante de 2° n'est couvert que par un point tous les 50cm. C'est insuffisant pour pouvoir mesurer une petite fracture, mais c'est suffisant pour mesurer un plan majeur affleurant sous la forme d'une face de trois mètres de côté.

Sous réserve que le problème de point de vue évoqué ci-dessus soit correctement géré, l'image solide permet de mesurer rapidement et précisément la position et l'orientation des discontinuités majeures. Le tableau 3.1 montre que cette technique est la plus

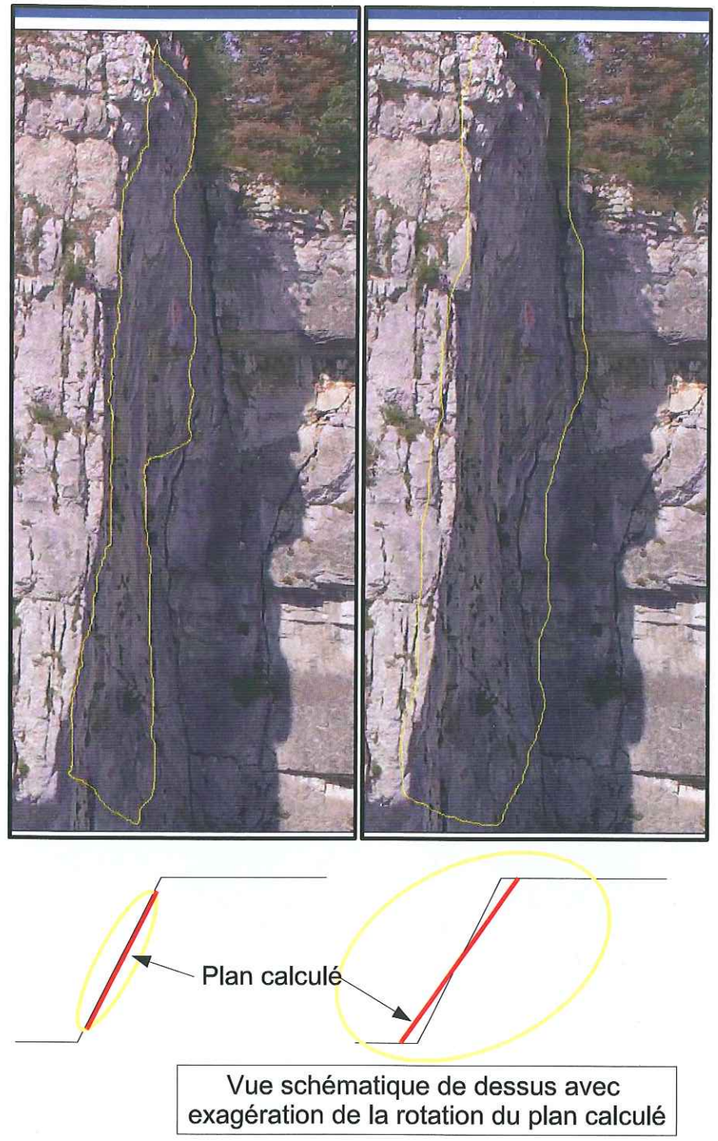


Fig. 3.20 – Mesure d'un plan de fracturation majeur via l'image solide, dans des conditions d'incidence rasante.
En jaune (en haut sur la photographie, en bas sur un schéma vu de dessus), la zone sélectionnée. En rouge la position du plan calculé. Il est plus difficile de sélectionner les pixels du plan, tous les pixels du plan et rien que les pixels du plan. Si des points sont sélectionnés de part et d'autre du plan (cas de droite), le plan calculé subira une rotation par rapport au plan réel.

rapide tout en étant aussi fiable que les autres.

3.4.3 Mesure sur le nuage de points ou le MNS

Nous avons vu en 3.2.3 que le principal problème pour la mesure directe des fractures sur le nuage de points était l'identification des plans de fracture. Ce problème ne se pose pas pour les discontinuités majeures qui sont des plans de taille suffisante pour être identifiés aisément sur le nuage de points texturé ou sur le modèle de surface (fig. 3.17).

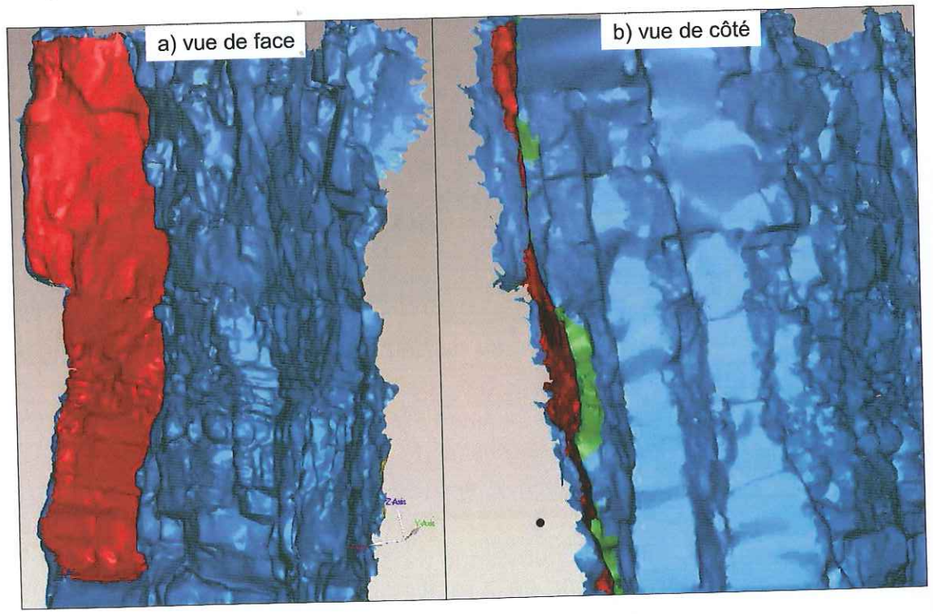


Fig. 3.21 – Sélection d'un plan de discontinuité majeur sur le modèle numérique de surface du Ravin de l'Aiguille.
Les triangles en rouge ont été sélectionnés depuis la vue de face comme faisant partie du plan. À partir de la vue de côté, on se rend compte que les triangles entourés n'en font pas partie.

La sélection des points faisant partie du plan est à peu près aussi rapide que sur

l'image solide (cf. tableau 3.1). Le problème du risque de sélection de points ne faisant pas partie du plan à calculer se pose de la même manière que lorsque la mesure est faite sur l'image solide, mais la possibilité de regarder le nuage de points depuis divers points de vue permet de mieux se rendre compte de l'appartenance au plan des points en bordure de celui-ci, et donc de ne pas sélectionner par erreur des points en limite du plan considéré (fig. 3.21). La méthode la plus rapide pour sélectionner la totalité d'un plan consiste à sélectionner l'ensemble du plan sur une vue face au plan, puis à regarder le même plan de côté pour désélectionner rapidement les triangles ou les points à l'extérieur du plan.

3.5 Stabilité actuelle et future

Le principal but recherché lors de l'étude des falaises supposées instables est d'être capable d'améliorer le diagnostic quant au risque d'éboulement, tant sur la probabilité d'un départ que sur la quantification du volume susceptible de partir. L'évaluation de cet aléa peut être fait par des méthodes très simples, comme celle de construction des pyramides de joints [Goodman et Shi, 1985], ou des méthodes plus compliquées telles que la simulation numérique.

3.5.1 Études analytiques d'après la position les discontinuités majeures

Ces études sont réalisées en modélisant la falaise par un ou quelques blocs de géométrie simple, considérés comme étant les volumes les plus susceptibles de se détacher. Les interactions entre le bloc et le reste de la paroi sont considérés comme des frottements plans sur plans avec un angle de frottement ϕ et une cohésion c. Le poids du

bloc s'applique en son centre de gravité.

Pour utiliser de telles méthodes, une hypothèse doit souvent être faite sur le pourcentage de ponts rocheux restant entre les faces en contact [Frayssines et Hantz, 2006]. En effet ce pourcentage conditionne la cohésion entre les deux surfaces. Régulièrement le frottement seul ne permet pas d'expliquer que l'escarpement soit encore en place alors qu'un pourcentage de ponts rocheux relativement faible permet de conclure à une stabilité durable de la zone.

En plus de la quantification des ponts rocheux, les données absolument nécessaires sont la position et l'orientation des fractures les plus importantes. Ces méthodes, suffisantes dans un premier temps, possède néanmoins deux travers qui, lors d'études plus poussés, leur font préférer les méthodes décrites dans les paragraphes suivants.

3.5.2 Études se basant sur les familles de fracture

Les plans de fracture qui vont effectivement jouer au moment de l'éboulement ne sont pas toujours visibles à l'affleurement. Un bloc limité par un plan non affleurant n'est pas modélisé par les méthodes du paragraphe précédent. Pour y remédier on peut réaliser des études qui se basent sur l'analyse statistique de l'orientation de la fracturation pour déterminer la forme des blocs susceptibles de se détacher. On détermine la forme de bloc la plus instable parmi tous les blocs dont les faces reprennent les orientations de fracture misent en lumière par l'étude statistique et on applique le calcul précédent pour cette forme de bloc.

3.5.3 Modélisation numérique

Il est enfin possible de prendre en compte l'ensemble des caractéristiques géométriques et mécaniques connues ou supposées de la paroi dans une simulation numérique. Aux échelles de temps qui nous intéressent, la roche peut être considérée comme un solide rigide-fragile pour lequel les ruptures n'ont lieu que sur des plans de fracture préexistants. Un code de calcul par élément distinct où chaque élément représente un bloc indéformable limité par des fractures, est donc à priori bien adapté à cette modélisation. Des codes tel que TRIDEC [Itasca Consulting Group, 1998] ou YADE [J. et F.V., ress] permettent ce type de modélisation. La limitation principale de ces codes réside dans le faible nombre de bloc qui peuvent être pris en compte. Afin de réduire le nombre de blocs indispensables pour la modélisation, il faut donc simplifier au maximum le MNS de la paroi.

3.6 Conclusion

Un des points clés d'une acquisition de données efficace est la question des points de vue, que ce soit pour une acquisition en lidar terrestre ou en photogrammétrie. Pour un lidar terrestre, il est important d'observer la scène depuis différents angles afin de ne pas avoir des orientations de fractures parallèles à la direction du rayon laser. Pour cela il faut généralement disposer d'un minimum de deux points de vue sur le site étudié. Les points de vue rasants sont à éviter autant que possible, la quantité de surface qui est masquée par les ressauts de la paroi devenant vite très important. Globalement on cherchera à se placer aussi près que possible pour améliorer la résolution et la précision des scannerisations.

En photogrammétrie il faudra trouver des points de vue permettant d'obtenir des

couples stéréoscopiques. Les différents points de vue seront, idéalement, sur un plan parallèle à la falaise étudiée, séparés les uns des autres d'une distance proche de celle les séparant de la falaise. Un bon équilibre est à trouver au sujet de cette distance : si les photographies sont prises de trop loin, la résolution et la précision seront dégradées, si elles sont prises de trop près, on ne bénéficiera pas d'une vue d'ensemble de la falaise et il faudra plusieurs couples stéréoscopiques différents pour étudier l'ensemble de la zone, posant dans ce cas le problème d'un modèle unique et la nécessité de recouvrir à une aérotriangulation.

En l'absence de points de vue terrestre de qualité, la question d'une acquisition aérienne se pose. Si la précision est similaire entre photogrammétrie terrestre et héliportée, les scannerisations laser héliportées sont d'une précision et d'une résolution un ordre de grandeur plus faible que ce que l'on peut obtenir en terrestre.

Si l'on compte utiliser une approche de type image solide, les photographies devront être prises depuis le même point de vue que la scannerisation. Cela évite de subir les effets gênants liés aux zones visibles depuis depuis l'un des points de vue et masquées depuis l'autre. De plus en plus, les scanners lidars sont équipés d'appareils photographiques numériques. La position et l'attitude de l'appareil photographique par rapport au laser est alors connue, ce qui dispense d'une orientation par photogrammétrie.

Chapitre 4

Conclusions et perspectives

Au terme de l'étude de ces différents sites par les méthodes de scannerisation lidar et de photogrammétrie, il est possible de conclure sur l'efficacité de ces deux méthodes par rapport à une étude classique. Les deux méthodes de télédétection étudiées permettent d'obtenir des données statistiques sur l'orientation des fractures de même qualités que les mesures in situ. Par rapport à une étude classique, il est plus facile d'obtenir une modélisation géométrique de l'ensemble de la paroi, avec une précision suffisante pour corriger les effets tridimensionnels lors des mesures géophysiques ou pour réaliser des simulation du comportement mécanique.

Avant de recourir à des méthodes de télédétection, il faut s'assurer de disposer de points de vue adéquats. Lorsque ces points de vue ne sont pas disponibles il est possible de réaliser des acquisitions aérienne. Si celles-ci ne détériorent pas la précision de la photogrammétrie, la précision d'une scannerisation laser héliporté est bien moins bonne que celle d'une acquisition terrestre. La précision obtenue au final reste suffisante pour l'étude de massifs où la fracturation découpe des blocs de dimension métrique à plurimétrique, mais ne permet pas de mesurer des plans de fracture de dimension

centimétrique à décimétrique.

Les deux méthodes ont chacune leurs avantages et leurs inconvénients. La photogrammétrie est pour le moment plus précise que la scannerisation lidar, et nécessite un temps d'intervention sur le terrain très réduit. La scannerisation lidar permet de disposer plus rapidement d'un modèle géométrique complet de la zone d'étude, et, une fois le modèle obtenu, de réaliser très rapidement des mesures ponctuelles.

Des développements restent à réaliser afin de tirer le meilleur parti possible de ces deux méthodes. Concernant la scannerisation laser et son exploitation par la technique de l'image solide, une amélioration du logiciel d'exploitation devra être réalisée à moyen terme. En dehors de la réalisations d'autres greffons pour des usages sépcifiques, cette amélioration passe aussi par une meilleure intégration des différents greffons développés dans un unique ensemble cohérent afin d'en faciliter l'usage pour l'utilisateur non spécialiste.

Un deuxième axe de développement consiste en la visualisation d'un couple stéréoscopique d'images solides. Ce développement devrait permettre de bénéficier en même temps de la vision stéréoscopique, point fort de la photogrammétrie, et de la faciliter de mesure qui découle de l'image solide. Un tel développement n'est vraisemblablement pas possible dans le cadre du logiciel ImageJ et nécessite de redévelopper une interface graphique complète.

Concernant la photogrammétrie, un travail reste à effectuer concernant l'utilisation dans le cadre de falaises instables de la corrélation automatique. La possibilité de réaliser une corrélation multiple à partir d'un grand nombre de photographie devrait permettre de rendre négligeable le nombre d'erreur de corrélation, principal inconvénient de la méthode, tout en améliorant la précision du modèle obtenu par rapport à l'utilisation d'un seul couple. Pour une utilisation terrestre le faible nombre de points de

vue utilisables risque de ne pas rendre cette méthode plus intéressante que la scannerisation laser, par contre pour une utilisation héliporté un tel système devrait permettre de combiner les avantage de la photogrammétrie exploitée manuellement (grande résolution et précision) et de la scannerisation lidar (facilité de mesure, connaissance de la position partout dans la zone d'étude).

Références

Abràmoff, M. D., P. J. Magalhães, et S. J. Ram (2004, jul). Image processing with imagej. *Biophoyonics International 11*(7), 36–42.

Bern, M. (2004). *Triangulations and mesh generation* (Second ed.)., Chapitre 25. CRC Press.

Besl, P. et H. McKay (1992, feb). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*(2), 239–256.

Bornaz, l. (2005). *L'analisi ed il trattamento dei dati laser scanner terrestri.* Ph. D. thesis, Politecnico di Milano.

Bornaz, L. et S. Dequal (2003). The solid image : a new concept and its applications. Dans *The international archive of the photogrammetry, remote sensing and spatial information sciences.*

Charles, P. (2001). Spécification technique principales caractéristiques du scanner soisic.

Chen, R.-F. (2007). com.pers.

Chen, Y. et G. Medioni (1992, April). Object modelling by registration of multiple range images. *Image and Vision Computing 10*(3), 145–155.

Cignoni, P., C. Montani, et R. Scopigno (1998, Feb). A comparaison of mesh simplification algorithms. *Computer & Graphics 22*(1), 37–54.

Deparis, J. (2007). *tude des éboulements rocheux par méthodes géophysiques.* Ph. D.

thesis, Université Joseph Fourier, Grenoble.

Dequal, S., A. Lingue, et F. Rinaudo (1996). Matching techniques and algorithms for some basic photogrammetric procedures in the low cost digital photogrammetric systems. Dans *The International Archives of Photogrammetry and Remote Sensing*.

Descartes, R. (1637). *La dioptrique*.

Dey, T. K., J. Giesen, et J. Hudson (2001). Delaunay based shape reconstruction from large data. Dans *Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*.

Dey, T. K. et S. Goswami (2006). Provable surface reconstruction from noisy samples. Dans *Special Issue on the 20th ACM Symposium on Computational Geometry*.

Egels, Y. (2002a). *Image acquisition. Physical aspect. Instruments*, Chapitre 1, pp. 1–77. Taylor & Francis.

Egels, Y. (2002b). *Mathematical model of the geometry of the aerial image*, Chapitre 1.1, pp. 2–15. Taylor & Francis.

Frayssines, M. et D. Hantz (2006). Failure mechanisms and triggering factors in calcareous cliffs of the subalpine ranges (french alps). *Engineering Geology 86*, 256–270.

Genty, H. (2002). Etudes en retour d'éboulements rocheux survenus sur les falaises calcaires de la région grenobloise. Mémoire de maîtrise stu, Université Joseph Fourier, Grenoble.

Goodman, R. et G. Shi (1985). *Block theory and its application to rock engineering*. United States : Prentice Hall,Englewood Cliffs, NJ.

Guering, J. (2001). Reliable 3d surface acquisition, registration and validation using statistical error models. Dans *Proceedings of the 3rd international conference on 3-D Digital Imaging and Modelling*.

Itasca Consulting Group, I. (1998). 3dec three-dimensional distinct element code

user's guide.

J., K. et D. F.V. (2008 in press). A new open-source software developed for numerical simulations using discrete modeling methods. *Computer Methods in Applied Mechanics and Engineering*.

Jeannin, M. (2005). *Etude des processus d'instabilités des versants rocheux par prospection géophysique. Apport du radar géologique*. Ph. D. thesis, Université Joseph Fourier, Grenoble.

Jongmans, D., J. D. T. Villemin, B. Fricout, A. Mathy, O.Meric, et L. Effendiantz (2007). Caractérisation multi-méthodes des aléas d'éboulements en masse.

Jung, F., F. Fuchs, et D. Boldo (2002). *Automatisation of aerotriangulation*, Chapitre 2.4, pp. 124–144. Taylor & Francis.

Kasser, M. (2002). *Digital image acquisition with airborne CCD cameras*, Chapitre 1.4, pp. 39–46. Taylor & Francis.

Linder, W. (2006). *Digital Photogrammetry A Practical Course* (second ed.). Berlin, Germany : Springer.

Linsen, L. (2001). Point cloud representation.

Natterer, F. (2001). *The mathematics of computerized tomography*. Philadelphia, PA, USA : Society for Industrial and Applied Mathematics.

Pauly, M., M. Gross, et L. Kobbelt (2002). Efficient simplification of point-sampled surfaces. Dans *Proceedings of the conference on Visualization '02*.

Rabbani, T., S. Dijkman, F. van den Heuvel, et G. Vosselman (2007, feb). An integrated approach for modelling and global registration of point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing 61*(6), 355–370.

riegldatasheet. Spécification technique du scanner lms z 420 i.

Sahbi, H., D. Jongmans, et R. Charlier (1997). Theoretical study of slope effects in resistivity surveys and applications. *Geophysical Prospecting 45*(5), 795–808.

Sithole, G. et G. Vosselman (2003). Comparison of filtering algorithms. Dans *Proceedings of the ISPRS working group III/3 workshop '3-D reconstruction from airborne laserscanner and InSAR data.*

Stenberg, L. (1996). the psd school.

Tarchi, D., N. Casagli, R. Fanti, D. D. Leva, G. Luzi, A. Pasuto, M. Pieraccini, et S. Silvano (2003, Feb). Landslide monitoring by using ground-based sar interferometry : an example of application to the tessina landslide in italy. *Engineering Geology 68*(1-2), 15–304.

Thom, C. (2002). *Radiometric and geometric precision*, Chapitre 1.9, pp. 63–77. Taylor & Francis.

Vallet, j. (2002). *Saisie de la couverture neigeuse des systèmes avalancheux par des systèmes aéroportés.* Ph. D. thesis, École Polytechnique Fédérale de Lausanne.

Wang, Y. (1998, jun). Principles and application of structural image matching. *ISPRS Journal of Photogrammetry and Remote Sensing 53*(3), 154–165.

Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. Dans *The Proceedings of the Seventh IEEE International Conference on Computer Vision.*

Annexes

.1 Certificat de calibration de la chambre UMK 13×18

04/02 2005 16:17 FAX 04 76 18 13 10 SINTEGRA 001/004

Sintégra — Géomètres-Experts

OPERATIONS FONCIERES
ETUDES V.R.D.

GEODESIE - PHOTOGRAMMETRIE - TOPOMETRIE GENERALE - METROLOGIE INDUSTRIELLE

TELECOPIE

De la part de	Michel MÉMIER
A l'attention de	Thierry Villemin - Université de Savoie
N° Fax destinataire	04 79 75 87 77
Nombre de pages	4 Y compris celle-ci
N/Référence	
Date	04.02.2005
Objet	

Ci-joint le calificat de Calibration de l'UMK.

Cordialement

Lucas

SARL au capital de 198000 € - R.C.S. GRENOBLE B 334 361 749 - N° D'INSCRIPTION OGE 28511

Siège social :
Avenue du Taillefer – BP 3 – 38241 MEYLAN Cedex
Tél. 04 76 18 13 13 - Fax : 04 76 18 13 10
e.mail : info@sintegra.fr

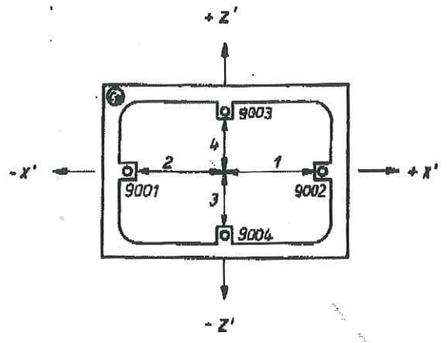
Bureau secondaire :
3 avenue Jules Ravat – 38500 VOIRON
Tél. : 04 76 05 82 31 - Fax : 04 76 05 58 81
e.mail : voiron@sintegra.fr

Certificat de calibrage

pour la

Chambre photogrammétrique universelle UMK 10/1318 U
Numéro 268 275 A

avec l'objectif Lamegon 8/100 B Numéro 73 87 972



Position des repères de cadre et distances de mesure dans le positif.

1. Distance principale pour un réglage de distance "∞" c_k = 99.29 mm

2. Position du point principal du cliché par rapport au centre des coordonnées d'image pour un réglage de la distance "∞"
   x' = ± 0.01 mm
   y'/z' = ± 0.00 mm

Remarque :

Les distances principales pour d'autres réglages de distances s'obtiennent par addition de la valeur Δc_k enregistrée sur le cliché à la valeur c_k citée ci-haut. Les valeurs Δc_k sont indiquées à ± 0,02 mm près.
La position invariable des points de référence du cliché est assurée à ± 0,02 mm lorsqu'on passe à d'autres réglages de distances.

3. Position des repères de cadre par rapport au centre des coordonnées d' image.

Repère de cadre	x' (mm)	z' (mm)
9001	- 80.387	± 0.000
9002	+ 80.173	± 0.000
9003	+ 0.002	+ 57.247
9004	- 0.002	- 57.288

4. Distorsion radiale $\Delta r'$ en µ pour un réglage de distance "∞".

r'_∞ (mm)	Distance de mesure 1	2	3	4	Valeur moyenne
15	+ 4	+ 5	+ 4	+ 4	+ 4
25	+ 6	+ 8	+ 6	+ 6	+ 7
40	+ 3	+ 7	+ 5	+ 4	+ 5
50	- 1	+ 5	+ 3	+ 2	+ 2
70	-12	+ 2			- 5

Pour passer à d'autres réglages de distance, les valeurs r' pour "∞" doivent être majorées de $\Delta\Delta r'$ selon la table 5.
Les valeurs de distorsion indiquées dans la table 5. (en µ) se réfèrent à des angles d'images fixes. Les rayons d'image associés varient donc en fonction de la focalisation.
Les rayons sont calculés de la manière suivante:

$$r'_e = r'_\infty \left(1 + \frac{z'}{g_4}\right) \quad \text{(mm)}$$

e	(m)	25	12	8	6	5	4,2	3,6	3,2	2,8
z'	(mm)	0,4	0,8	1,2	1,6	2,0	2,4	2,8	3,2	3,6
e	(m)	2,6	2,3	2,1	2,0	1,8	1,7	1,6	1,5	1,4
z'	(mm)	4,0	4,4	4,8	5,2	5,6	6,0	6,4	6,8	7,2

3

5. Table des majorations $\Delta\Delta r'$ à la distorsion radiale pour d'autres réglages de distance.

Lamegon $\tilde{8}$/100 B

r'_{oo}	25	12	8	6	5	4,2	3,6	3,2	2,8
15	0	-1	-1	-1	-2	-2	-3	-3	-3
25	0	-1	-2	-2	-3	-3	-4	-4	-5
40	-1	-1	-2	-2	-3	-4	-4	-5	-6
50	0	-1	-1	-2	-2	-3	-3	-4	-4
70	0	0	0	+1	+1	+1	+1	+1	+1

r'_{oo}	2,5	2,3	2,1	2	1,8	1,7	1,6	1,5	1,4
15	-4	-4	-4	-5	-5	-6	-6	-7	-7
25	-6	-6	-6	-7	-8	-8	-9	-9	-10
40	-6	-7	-7	-8	-9	-9	-10	-11	-11
50	-5	-5	-6	-6	-7	-7	-8	-8	-9
70	+2	+2	+2	+2	+2	+2	+2	+3	+3

.2 Certificat de calibration de l'appareil kodak DCS pro

POLITECNICO DI TORINO

Prof. Fulvio RINAUDO

Certificat de calibration

Type de camera: kodak DCSproSLR Date: 30/05/2005

Nom du travail: kodakDCSproSLR 24mm_filtro
Type de camera: kodak DCSproSLR
N° de série de la camera:
Type d'objectif: Nikkor 24mm filtro UV52mm
N° de série de l'objectif:
Distance de mise à feu: infinit
Distance iperfocale: - m
Ouverture du diaframme: 22
Dimensions du pixel: 8 micron

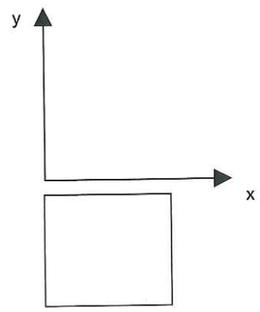
Paramètres de calibration

Paramètres d'orientation internes

Distance principale: c= 24.257 mm
Coordonnées du point principal: x0= 18.023 mm
y0= -11.888 mm

Paramètres de distortion

k1= -1.61536142956465E-04 1/mm^2
k2= 2.52972731477173E-07 1/mm^4



Dipartimento di Ingegneria del Territorio, dell'Ambiente e delle Geotecnologie

Politecnico di Torino – Dipartimento di Ingegneria del Territorio, dell'Ambiente e delle Geotecnologie (DITAG)
Corso Duca degli Abruzzi, 24 – 10129 Torino Italia
tel: +39 011 564 7659 fax: +39 011 564 7699
e-mail: fulvio.rinaudo@polito.it url: www.polito.it/ricerca/dipartimenti/ditag/

POLITECNICO DI TORINO

Prof. Fulvio RINAUDO

Valeurs de la distorsion de l'objectif en fonction de la distance radiale r en direction de la diagonale (micron)

r (mm)	Dist x	Dist y	Dist
1	-.1	-.0	-.1
2	-1.0	-.7	-1.2
3	-3.5	-2.3	-4.3
4	-8.3	-5.5	-10.0
5	-16.1	-10.7	-19.4
6	-27.3	-18.2	-32.9
7	-42.5	-28.3	-51.1
8	-61.9	-41.2	-74.4
9	-85.5	-57.0	-102.8
10	-113.3	-75.5	-136.2
11	-144.9	-96.6	-174.2
12	-179.8	-119.9	-216.1
13	-217.1	-144.7	-260.9
14	-255.6	-170.4	-307.2
15	-293.7	-195.8	-353.0
16	-329.8	-219.8	-396.3
17	-361.4	-240.9	-434.4
18	-386.1	-257.4	-464.0
19	-400.7	-267.1	-481.5
20	-401.6	-267.7	-482.7
21	-385.0	-256.7	-462.8
22	-346.3	-230.9	-416.3

Dipartimento di Ingegneria del Territorio, dell'Ambiente e delle Geotecnologie

Politecnico di Torino – Dipartimento di Ingegneria del Territorio, dell'Ambiente e delle Geotecnologie (DITAG)
Corso Duca degli Abruzzi, 24 – 10129 Torino Italia
tel: +39 011 564 7659 fax: +39 011 564 7699
e-mail: fulvio.rinaudo@polito.it url: www.polito.it/ricerca/dipartimenti/ditag/

.3 T2IS

```
#!/bin/perl
##on suppose la surface de format :
## x y z x y z x y z (un triangle par ligne
##les paramètre d'orientation/calidration de la photo donnés dans un fichier qvb :
##focale(mm) mat1-1 mat1-2 mat1-3 mat2-1 mat2-2 mat2-3 mat3-1 mat3-2 mat 3-3 deltaX deltaY deltaZ
largeur(pix) hauteur orientationinterne1-1(inclus nbdepix/mm) orientationinterne1-2 centrefocalx
orientationinterne2-1 orientationinterne2-2 centrefocaly
##les trois fichiers de coordonnées des pixels, directement ouvrable par imageJ
open(INFILE,  "photocoord.txt" ) or die "impossible d'ouvrir photocoord.txt";#le fichier
d'orientationcalibration
open(INFILE2,  "rocher.txt" ) or die "impossible d'ouvrir surface.txt";#le fichier de triangles
my (@ligne, $i, $x1, $y1, $z1, $x2, $z2, $y2, $x3, $y3, $z3, $xp1 ,$yp1 ,$xp2 ,$yp2, $xp3, $yp3, $xp, $yp,
$zp, $lectu, $j, @photocoord, @rayon, $temp, $xmin, $xmax, $ymin, $ymax, $temp3);
$lectu=<INFILE>;#zap la première ligne d'en-tête.
$lectu=<INFILE>;
@photocoord=(split ' ', $lectu );
#initialisont  @rayon
$compteur=0;
open(OUTFILEr,  ">photor.txt" ) or die "impossible d'ouvrir photox.txt" ;

@rayon =(split ' ', $lectu );
for ($i=1; $i<= $photocoord[13]; $i++){
    $temp=($i-1)*$photocoord[14];
for($j=1;$j<=$photocoord[14]; $j++){
        $temp2=$j+$temp;
$rayon[$temp2]=-100000;
    }

}

$lectu=<INFILE2>;
@ligne=(split ' ', $lectu );



while($lectu){
@ligne=(split ' ', $lectu );#lecture du premier triangle
$lectu=<INFILE2>;


 #calcul des coordonnées pixels des sommets du triangle
 #il faudrait ajouter la distortion radiale (2ième temps)
    $x1 = $photocoord[1]*($ligne[0]-$photocoord[10])+$photocoord[2]*($ligne[1]-$photocoord[11])+$photocoord[3]*
($ligne[2]-$photocoord[12]);
    $y1 = $photocoord[4]*($ligne[0]-$photocoord[10])+$photocoord[5]*($ligne[1]-$photocoord[11])+$photocoord[6]*
($ligne[2]-$photocoord[12]);
    $z1 = $photocoord[7]*($ligne[0]-$photocoord[10])+$photocoord[8]*($ligne[1]-$photocoord[11])+$photocoord[9]*
($ligne[2]-$photocoord[12]);
    $x1 = $x1 /$z1;
    $y1 = $y1 /$z1;
    $xp1 = -$x1*$photocoord[0];
    $yp1 = -$y1*$photocoord[0];
    $temp = $xp1*$photocoord[15]+ $yp1*$photocoord[16]+ $photocoord[17];
    $yp1 = $xp1*$photocoord[18]+ $yp1*$photocoord[19]+ $photocoord[20];
    $xp1=$temp;
   $x2 = $photocoord[1]*($ligne[3]-$photocoord[10])+$photocoord[2]*($ligne[4]-$photocoord[11])+
$photocoord[3]*($ligne[5]-$photocoord[12]);
    $y2 = $photocoord[4]*($ligne[3]-$photocoord[10])+$photocoord[5]*($ligne[4]-$photocoord[11])+
$photocoord[6]*($ligne[5]-$photocoord[12]);
    $z2 = $photocoord[7]*($ligne[3]-$photocoord[10])+$photocoord[8]*($ligne[4]-$photocoord[11])+
$photocoord[9]*($ligne[5]-$photocoord[12]);
    $x2 = $x2 /$z2;
    $y2 = $y2 /$z2;
    $xp2 = -$x2*$photocoord[0];
```



```
    $yp2 = -$y2*$photocoord[0];
    $temp = $xp2*$photocoord[15]+ $yp2*$photocoord[16]+ $photocoord[17];
    $yp2 = $xp2*$photocoord[18]+ $yp2*$photocoord[19]+ $photocoord[20];
    $xp2=$temp;
    $x3 = $photocoord[1]*($ligne[6]-$photocoord[10])+$photocoord[2]*($ligne[7]-$photocoord[11])+
$photocoord[3]*($ligne[8]-$photocoord[12]);
    $y3 = $photocoord[4]*($ligne[6]-$photocoord[10])+$photocoord[5]*($ligne[7]-$photocoord[11])+
$photocoord[6]*($ligne[8]-$photocoord[12]);
    $z3 = $photocoord[7]*($ligne[6]-$photocoord[10])+$photocoord[8]*($ligne[7]-$photocoord[11])+
$photocoord[9]*($ligne[8]-$photocoord[12]);
    $x3 = $x3 /$z3;
    $y3 = $y3 /$z3;
    $xp3 = -$x3*$photocoord[0];
    $yp3 = -$y3*$photocoord[0];
    $temp = $xp3*$photocoord[15]+ $yp3*$photocoord[16]+ $photocoord[17];
    $yp3 = $xp3*$photocoord[18]+ $yp3*$photocoord[19]+ $photocoord[20];
    $xp3=$temp;
  # calcul de xmin, xmax, ymin ymax
    if ($xp1<$xp2){
        $xmin=$xp1;
        $xmax=$xp2;
    }
    else{
        $xmin=$xp2;
        $xmax=$xp1;
    }
    if ($xp3<$xmin){
        $xmin=$xp3;
    }
    if ($xp3>$xmax){
        $xmax=$xp3;
    }
    if ($yp1<$yp2){
        $ymin=$yp1;
        $ymax=$yp2;
    }
    else{
        $ymin=$yp2;
        $ymax=$yp1;
    }
    if ($yp3<$ymin){
        $ymin=$yp3;
    }
    if ($yp3>$ymax){
        $ymax=$yp3;
    }
    if ($xmin<1) {
        $xmin=1;
    }
    if ($ymin<1) {
        $ymin=1;
    }
    if ($xmax>$photocoord[13]) {
        $xmax=$photocoord[13];
    }
    if ($ymax>$photocoord[14]) {
        $ymax=$photocoord[14];
    }
      #si les pixels sont bien sur la photo
     if(($xmax>1) && ($ymax>1) &&($xmin<$photocoord[13])  && ($ymin<$photocoord[14])){
         #boucle sur les pixels concernés
         for ($i= int($xmin)-1; $i<=$xmax; $i++){
            for   ($j= int($ymin)-1; $j<=$ymax; $j++){#tester si le pixel est bien dans le triangle et
calculer l'intersection

                $temp = ($xp2-$i)*($yp3-$j)-($yp2-$i)*($xp3-$j);
                if($temp!=0){#si le point n'est pas sur (BC)
                        $temp3=(($xp1-$i)*($yp2-$j)-($yp1-$j)*($xp2-$i))/$temp;
```

```
if ($temp3>0){
$temp=-(($xp1-$i)*($yp3-$j)-($yp1-$j)*($xp3-$i))/$temp;
if ($temp>0){#les 3 paramètre du barycentre sont >0 dc le pt est ds le triangle
                                $zp=($z1 + $temp*$z2 + $temp3*$z3)/(1 + $temp + $temp3);#le projeté du
barycentre reste barycentre
                                $temp=($i-1)*$photocoord[14];
        $temp2=$j+$temp;
if ($rayon[$temp2]<$zp){


$rayon[$temp2]=$zp;


}
                        }
                    }
                }
       else{
        $temp=($i-$xp2)/($xp3-$i);
        if($temp>=0){
$zp=($z2+$temp*$z3)/(1+$temp);
        $temp=($i-1)*$photocoord[14];
         $temp2=$j+$temp;
if ($rayon[$temp2]<$zp){


$rayon[$temp2]=$zp;


}
       }
}
       }
      }
         }
        }
    }
}


#plus qu'à imprimer tout ca.
open(OUTFILE,  ">photox.txt" ) or die "impossible d'ouvrir photox.txt" ;
open(OUTFILE2,  ">photoy.txt" ) or die "impossible d'ouvrir photoy.txt" ;
open(OUTFILE3,  ">photoz.txt" ) or die "impossible d'ouvrir photoz.txt" ;

for ($j=$photocoord[14] ;$j>0;$j--){
    $temp=$j;
for($i=1;$i<=$photocoord[13];$i++){#on remet dans l'ancien système de coordonnées
$temp2=$temp+($i-1)*$photocoord[14];

if ($rayon[$temp2]>-100000){
$xp = ($i-$photocoord[17])*$photocoord[15]- ($j-$photocoord[20])*$photocoord[18];
         $yp = -($i-$photocoord[17])*$photocoord[16]+ ($j-$photocoord[20])*$photocoord[19];
         $zp = $rayon[$temp2];
         $xp = -$xp * $zp /  $photocoord[0]/($photocoord[15]*$photocoord[19]-$photocoord[16]*
$photocoord[18]);
         $yp = -$yp * $zp / $photocoord[0]/($photocoord[15]*$photocoord[19]-$photocoord[16]*
$photocoord[18]);
         $x1 = $photocoord[1] * $xp + $photocoord[4] * $yp + $photocoord[7] * $zp + $photocoord[10];
         $y1 = $photocoord[2] * $xp + $photocoord[5] * $yp + $photocoord[8] * $zp + $photocoord[11];
         $z1 = $photocoord[3] * $xp + $photocoord[6] * $yp + $photocoord[9] * $zp + $photocoord[12];
        #on imprime x dans le fichierX, Y dans le fichier Y et Z dans le fichierZ
         print (OUTFILE $x1);
         print (OUTFILE "\t");
         print (OUTFILE2 $y1);
         print (OUTFILE2 "\t");
         print (OUTFILE3 $z1);
         print (OUTFILE3 "\t");
 print (OUTFILEr $rayon[$temp2]);
```



```
print (OUTFILEr "\n");
}
else{
         print (OUTFILE 0);
         print (OUTFILE "\t");
         print (OUTFILE2 0);
         print (OUTFILE2 "\t");
         print (OUTFILE3 0);
         print (OUTFILE3 "\t");
}
}
print (OUTFILE "\n");
print (OUTFILE2 "\n");
print (OUTFILE3 "\n");
}
```

.4 ImageSolide

```
import ij.*;
import ij.text.*;
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.io.*;
import java.util.*;
import java.awt.datatransfer.*;
import ij.plugin.*;
import ij.io.*;


/**
normalement le plugin principal qui appelle les autres
   */
public class ImageSolide_ implements PlugIn, MouseListener, MouseMotionListener, KeyListener {
ImagePlus imagesolide;
ImagePlus stereogramme;
    TextWindow planmoyen;
    ImageStack IS;
ImageCanvas canvas;
ImageStack IS2;
ImageCanvas canvas2;
boolean color;//RGB24BIT ou gris32bit (le codage des coordonnées est différents
boolean aff_dist;//mode coordonnées ou écart
boolean bla;
static Vector images = new Vector();
int X1;//coordonnées pixels du point de référence en mode écart
int Y1;


public void run(String arg) {

     if(!openimagesolide(arg)){//si il n'y a pas d'image solide on ne fait rien
        return;
    }
     if(!openstereogramme(arg)){//si il n'y a pas de stéréogramme, on en crée un vide
        createstereogramme();
    }
     if(!openplanmoyen(arg)){//si il n'y a pas de fichier texte des plans moyens, on en crée un vide.
        createplanmoyen();
    }
    bla=false;
    color=false;
    aff_dist = false;//mode coordonnées
    IJ.register(ImageSolide_.class);
    ImageProcessor ip=imagesolide.getProcessor();
    if (ip instanceof ColorProcessor){
        color=true;
    }
 Integer id = new Integer(imagesolide.getID());
 if (images.contains(id)) {
    IJ.log("Already listening to this image");//empêche d'avoir plusieurs mouse_listener sur la même image
    return;
 } else {
    ImageWindow win = imagesolide.getWindow();
    canvas = win.getCanvas();
    canvas.addMouseListener(this);
    canvas.addMouseMotionListener(this);
    images.addElement(id);
        canvas.addKeyListener(this);
```



```
       id = new Integer(stereogramme.getID());
       if (images.contains(id)) {
      IJ.log("Already listening to this image");
      return;
   } else {
       win = stereogramme.getWindow();
       canvas2 = win.getCanvas();
       canvas2.addMouseListener(this);
       canvas2.addMouseMotionListener(this);
       canvas2.addKeyListener(this);
       images.addElement(id);
      }
}
}

 public boolean openimagesolide(String arg){//on ouvre l'image solide
      OpenDialog od = new OpenDialog("Open imagesolide...", arg);
      String directory = od.getDirectory();
      String name = od.getFileName();
      if (name==null)
          return false;
      Opener op=new Opener();
      imagesolide= op.openImage(directory,name);
      IS = imagesolide.getStack();
      imagesolide.show();
      return true;//il faudrait ajouter une vérif comme quoi c'est bien une image solide
 }
 public boolean openstereogramme(String arg){//on ouvre le stéréogramme
      OpenDialog od = new OpenDialog("Open stereogramme...", arg);
      String directory = od.getDirectory();
      String name = od.getFileName();
      if (name==null)
          return false;
      Opener ope=new Opener();
      stereogramme= ope.openImage(directory,name);
      IS2 = stereogramme.getStack();
      stereogramme.show();
      return true;
  }
  public boolean openplanmoyen(String arg){//on ouvre le fichier texte "planmoyen"
      OpenDialog od = new OpenDialog("Open imagesolide...", arg);
      String directory = od.getDirectory();
      String name = od.getFileName();
      if (name==null)
          return false;
      String path = directory+name;
      planmoyen=new TextWindow(path, 500, 500);
      return true;
  }
public void createstereogramme(){//on crée le stéréogramme
      int w = 501, h = 501;
   ImageProcessor ip = new ColorProcessor(w, h);//on crée donc un nouveau colorprocessor
      int[] pixels = (int[])ip.getPixels();
      for (int i=0; i<501*501;i++){
            pixels[i]=-1;  //on initialise tous les pixels sur blanc
      }
      for (int x = 0; x < 251; x++) {//on trace le cercle de rayon 250 pixel en noir
           int y = (int) (Math.sqrt((500-x)*x));
           int i = (250-y)*501+x;
           pixels[i]=0;
           i = (250+y)*501+x;
           pixels[i]=0;
           i = (250-y)*501+500-x;
           pixels[i]=0;
           i = (250+y)*501+500-x;
           pixels[i]=0;
      }
      for (int y = 0; y < 251; y++) {//on trace le cercle de rayon 250 pixel en noir
```

```
            int x = (int) (Math.sqrt((500-y)*y));
            int i = (250-x)+y*501;
            pixels[i]=0;
            i = (250+x)+y*501;
            pixels[i]=0;
            i = (250-x)+(500-y)*501;
            pixels[i]=0;
            i = (250+x)+(500-y)*501;
            pixels[i]=0;
        }
        stereogramme=new ImagePlus("stereogramme.tif", ip);
        stereogramme.show();//on crée l'image du stéréogramme et on l'affiche
        IS2 = stereogramme.getStack();

 }
public void createplanmoyen(){
  planmoyen= new TextWindow("planmoyen","a\tb\tc\td\tdip\tdip direction\t","", 500,500);



 }
 public void mousePressed(MouseEvent e) {//quand la souris est pressée, on affiche les coordonnées
 X Y Z du point
 int x = e.getX();
 int y = e.getY();
     int offscreenX = canvas.offScreenX(x);
 int offscreenY = canvas.offScreenY(y);
     float temp=0;

     String val=" ";
     for (int n=2; n<5; n++) {
         ImageProcessor ip =IS.getProcessor(n);
         if (color){
             temp= ip.getPixel(offscreenX,offscreenY);

             temp=temp/1000+8000;
         }
         else{
             temp= Float.intBitsToFloat(ip.getPixel(offscreenX,offscreenY));
         }
         val=val+" "+IJ.d2s(temp,3);


     }
     IJ.showStatus  (val);
   }

 public void mouseReleased(MouseEvent e) {//doit être présent dans les MouseListener
 }

 public void mouseDragged(MouseEvent e) {//doit être présent dans les MouseListener

 }

 public static String modifiers(int flags) {
 String s = " [ ";
 if (flags == 0) return "";
 if ((flags & Event.SHIFT_MASK) != 0) s += "Shift ";
 if ((flags & Event.CTRL_MASK) != 0) s += "Control ";
 if ((flags & Event.META_MASK) != 0) s += "Meta (right button) ";
 if ((flags & Event.ALT_MASK) != 0) s += "Alt ";
 s += "]";
 if (s.equals(" [ ]"))
   s = " [no modifiers]";
 return s;
 }

 public void mouseExited(MouseEvent e) {}//doit être présent dans les MouseListener
```



```
public void mouseClicked(MouseEvent e) {//quand on clique
    int x = e.getX();
int y = e.getY();

ImageWindow win = stereogramme.getWindow();
    if(win==WindowManager.getCurrentWindow()){//si l'image courante est le stéréogramme
        int offscreenX = canvas2.offScreenX(x);
    int offscreenY = canvas2.offScreenY(y);
        ColorProcessor ip =(ColorProcessor)IS2.getProcessor(1);

        int color= ip.getPixel(offscreenX,offscreenY);//on récupère la couleur du point cliqué
        if(color<0){
            color=color+16777216;
        }
        if(color!=16777215){ //si cette couleur n'est pas blanc,c'est qu'on a cliqué sur un plan
selectionné
            afficheROI(color); //on réaffiche la zone de sélection
            afficheequation(color);//et l'équation du plan
        }
        return;
    }
    //sinon l'image courante est l'image solide, on bascule entre les modes écarts avec le point
cliqué et coordonnées
int offscreenX = canvas.offScreenX(x);
int offscreenY = canvas.offScreenY(y);
    X1= offscreenX  ;
Y1= offscreenY ;
    aff_dist=!aff_dist;
    }
public void mouseEntered(MouseEvent e) {}//doit être présent dans les MouseListener
public void mouseMoved(MouseEvent e) {//quand la souris bouge, on affiche les coordonnées/la distance
    int x = e.getX();
int y = e.getY();
int offscreenX = canvas.offScreenX(x);//coordonnées pixel absolu (et non relative à la fenètre d'affichage)
int offscreenY = canvas.offScreenY(y);
String val=" ";
    float temp=0;
    float temp2=0;
    if (!aff_dist){//si on est en mode coordonnées

        for (int n=2; n<5; n++) {//pour les trois couches spatiales (X Y et Z)
            ImageProcessor ip =IS.getProcessor(n);
            if (color){//on calcule la valeur de la coordonnés (différement suivant que l'image
est RGB24BIT ou GRIS32BIT
                temp= ip.getPixel(offscreenX,offscreenY);

                temp=temp/1000+8000;
            }
            else{
                temp= Float.intBitsToFloat(ip.getPixel(offscreenX,offscreenY)) ;
            }
            val=val+" "+IJ.d2s(temp,3);//on ajoute la coordonnées dans val

        }
    }
    else{//si on est en mode écart
       double dist=0;
        ImageProcessor ip =IS.getProcessor(2); //pour la couches X
        if (color){ //on calcule la valeur de la coordonnés (différement suivant que l'image est
RGB24BIT ou GRIS32BIT)
            temp= ip.getPixel(offscreenX,offscreenY);//pour la position actuelle
            temp2= ip.getPixel(X1,Y1);//et la position de référence

             temp=temp/1000+8000;
             temp2=temp2/1000+8000;
        }
        else{
```

```
      temp= Float.intBitsToFloat(ip.getPixel(offscreenX,offscreenY));
      temp2= Float.intBitsToFloat(ip.getPixel(X1,Y1));
}
temp=temp-temp2; //donc on calcule l'écart entre les 2
val="dx= "+IJ.d2s(temp,3);
dist=temp*temp;  //on ajoute dans dist l'écart au carré
ip =IS.getProcessor(3);//pour la couche Y même chose
if (color){
    temp= ip.getPixel(offscreenX,offscreenY);
    temp2= ip.getPixel(X1,Y1);

     temp=temp/1000+8000;
     temp2=temp2/1000+8000;
}
else{
        temp= Float.intBitsToFloat(ip.getPixel(offscreenX,offscreenY));
        temp2= Float.intBitsToFloat(ip.getPixel(X1,Y1));
}
temp=temp-temp2;
val=val+" dy= "+IJ.d2s(temp,3);
dist=dist+temp*temp;
ip =IS.getProcessor(4);//pour la couche Z même chose
if (color){
    temp= ip.getPixel(offscreenX,offscreenY);
    temp2= ip.getPixel(X1,Y1);

     temp=temp/1000+8000;
     temp2=temp2/1000+8000;
}
else{
        temp= Float.intBitsToFloat(ip.getPixel(offscreenX,offscreenY));
        temp2= Float.intBitsToFloat(ip.getPixel(X1,Y1));
}
temp=temp-temp2;
        temp2= Float.intBitsToFloat(ip.getPixel(X1,Y1));
      }
      temp=temp-temp2;
      val=val+" dz= "+IJ.d2s(temp,3);
      dist=dist+temp*temp;
      dist= Math.sqrt(dist); //plus qu'à prendre la racine pour avoir la distance entre les points
      val=val+" distance= "+IJ.d2s(dist,3);
   }
   IJ.showStatus  (val);//et on affiche val(qui contient soit les coordonnées, soit les écarts)
dans la barre de status d'imageJ
}

public void keyPressed(KeyEvent e) {//quand une touche est pressée
  int key = e.getKeyCode();
  IJ.showStatus(" "+key );//on affiche discrètement le numéro de la touche, parce que je connais pas le
code ascii par coeur et que ca me fait gagner du temps quand je veux rajouter un nouveau plugin sur une touche

  if (key==66){ //si la touche est b
     Object thePlugIn=IJ.runPlugIn("planmoyen_","");  //on execute le plugin planmoyen
     String directory = OpenDialog.getDefaultDirectory();
     String name = imagesolide.getTitle();
     name = name+"plan.txttemp";
     try {   //on lit dans le fichier temporaire l'équation du plan qui vient d'être calculée
  BufferedReader r = new BufferedReader(new FileReader(directory + name));
  planmoyen.load(r);//et on l'ajoute au texte(ouvert) planmoyen
  r.close();
   }
  catch (Exception er) {
IJ.error(er.getMessage());
  }
   }

   if (key==71){//si la touche est g
      Object thePlugIn=IJ.runPlugIn("face_","");
   }
   if (key==84){//si la touche est t
      Object thePlugIn=IJ.runPlugIn("listing_","");
```



```
   }
   if (key==89){//si la touche est y
      Object thePlugIn=IJ.runPlugIn("trouveintersect_","");
   }
   if (key==85){ //si la touche est u

       Object thePlugIn=IJ.runPlugIn("modelesimplifie_","");
    }
}

public void keyReleased (KeyEvent e) {}
public void keyTyped (KeyEvent e) {

    }


    void showAbout( int j ) {//me sert pour afficher des resultats intermédiaire pendant le développement
IJ.showMessage("listing_...",
j+"points ont été sauvegardés\n"+
"sur un total de pixels" );
}


  public void afficheROI( int color){//affiche la zone de sélection ayant permit de trouver l'équation
du plan numéro "color"
    String directory = OpenDialog.getDefaultDirectory();
    String name = imagesolide.getTitle();
    name = name+"plan.txt"+"temproi"+color;//le nom du fichier d'enregistrement de la selection
    WindowManager.setTempCurrentImage(imagesolide); //on met l'image solide en image courante
    try { //on ouvre le fichier de sélection, ce qui reselectionne la zone en question
        openRoi( directory , name);
    }
    catch (IOException e) {
  String msg = e.getMessage();
  if (msg==null || msg.equals(""))
    msg = ""+e;
   IJ.error("ROI Reader", msg);
}
    WindowManager.setTempCurrentImage(imagesolide);
  }

public void openRoi(String dir, String name) throws IOException {
String path = dir+name;
RoiDecoder rd = new RoiDecoder(path);//on lit le fichier
Roi roi = rd.getRoi();
Rectangle r = roi.getBounds();
ImagePlus img = WindowManager.getCurrentImage();
if (img==null || img.getWidth()<(r.x+r.width) || img.getHeight()<(r.y+r.height)) {//normalement
n'arrive pas! le ROI devrait être dans l'image
ImageProcessor ip =  new ByteProcessor(r.x+r.width+10, r.y+r.height+10);
ip.setColor(Color.white);
ip.fill();
img = new ImagePlus(name, ip);
img.show();
}
img.setRoi(roi);
}
   public void afficheequation( int color){

    TextPanel texte =planmoyen.getTextPanel();
    String plan = texte.getLine(color);
    IJ.showStatus(plan);//on affiche la ligne n° "color" du texte planmoyen

 }
```

}

.5 MouseListenerb

```
import ij.*;
import ij.plugin.filter.PlugInFilter;
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

/**
This plugin implements the MouseListener and MouseMotionListener interfaces
and listens for mouse events generated by the current image.
*/
public class Mouse_Listenerb_ implements PlugInFilter, MouseListener, MouseMotionListener {
ImagePlus img;
    ImageStack IS;
ImageCanvas canvas;
boolean color;//image couleur ou non
boolean aff_dist;//bascule coordonnées du point/écart avec un point.
static Vector images = new Vector();
int X1;//coordonnées pixel du point avec lequel on mesure la distance
int Y1;

public int setup(String arg, ImagePlus img) {
    aff_dist = false;
    this.img = img;
IS = img.getStack();
IJ.register(Mouse_Listenerb_.class);
return DOES_ALL+NO_CHANGES;
}

public void run(ImageProcessor ip) {
    color=false;
    if (ip instanceof ColorProcessor){
      color=true;
    }
Integer id = new Integer(img.getID());
if (images.contains(id)) {
IJ.log("Already listening to this image");//empêche d'avoir plusieurs mouse_listener sur la même image
return;
} else {
ImageWindow win = img.getWindow();
canvas = win.getCanvas();
canvas.addMouseListener(this);
canvas.addMouseMotionListener(this);
images.addElement(id);
}
}

public void mousePressed(MouseEvent e) {//quand la souris est pressée, on affiche les coordonnées
X Y Z du point
int x = e.getX();
int y = e.getY();
int offscreenX = canvas.offScreenX(x);
int offscreenY = canvas.offScreenY(y);
    float temp=0;
    String val=" ";
    for (int n=2; n<5; n++) {
        ImageProcessor ip =IS.getProcessor(n);
        if (color){
            temp= ip.getPixel(offscreenX,offscreenY);

            temp=temp/1000+8000;
        }
        else{
```

```
            temp= Float.intBitsToFloat(ip.getPixel(offscreenX,offscreenY));
        }
        val=val+" "+temp;

    }
    IJ.showStatus  (val);
}

public void mouseReleased(MouseEvent e) {//doit être présent dans les MouseListener
}

public void mouseDragged(MouseEvent e) {//doit être présent dans les MouseListener

}

public static String modifiers(int flags) {//doit être présent dans les MouseListener
String s = " [ ";
if (flags == 0) return "";
if ((flags & Event.SHIFT_MASK) != 0) s += "Shift ";
if ((flags & Event.CTRL_MASK) != 0) s += "Control ";
if ((flags & Event.META_MASK) != 0) s += "Meta (right button) ";
if ((flags & Event.ALT_MASK) != 0) s += "Alt ";
s += "]";
if (s.equals(" [ ]"))
  s = " [no modifiers]";
return s;
}

public void mouseExited(MouseEvent e) {} //doit être présent dans les MouseListener

public void mouseClicked(MouseEvent e) {//quand on clique, on bascule de coordonnées à distance avec
le point cliqué
    int x = e.getX();
int y = e.getY();
int offscreenX = canvas.offScreenX(x);
int offscreenY = canvas.offScreenY(y);
    X1= offscreenX  ;
Y1= offscreenY ;
    aff_dist=!aff_dist;
    }
public void mouseEntered(MouseEvent e) {}//doit être présent dans les MouseListener
public void mouseMoved(MouseEvent e) {//quand la souris bouge, on affiche les coordonnées/la distance
    int x = e.getX();
int y = e.getY();
int offscreenX = canvas.offScreenX(x);//coordonnées pixel absolu (et non relative à la fenêtre d'affichage)
int offscreenY = canvas.offScreenY(y);
String val=" ";
    float temp=0;
    float temp2=0;
    if (!aff_dist){//si on est en mode coordonnées

        for (int n=2; n<5; n++) { //pour les trois couches spatiales (X Y et Z)
            ImageProcessor ip =IS.getProcessor(n);
            if (color){ //on calcule la valeur de la coordonnés (différement suivant que l'image est
 RGB24BIT ou GRIS32BIT
                temp= ip.getPixel(offscreenX,offscreenY);

                temp=temp/1000+8000;
            }
            else{
                temp= Float.intBitsToFloat(ip.getPixel(offscreenX,offscreenY)) ;
            }
            val=val+" "+temp;//on ajoute la coordonnées dans val

        }
    }
    else{//si on est en mode écart
       double dist=0;     //pour la couches X
```



```
       ImageProcessor ip =IS.getProcessor(2);
       if (color){//on calcule la valeur de la coordonnés (différement suivant que l'image est
RGB24BIT ou GRIS32BIT)
           temp= ip.getPixel(offscreenX,offscreenY);//pour la position actuelle
           temp2= ip.getPixel(X1,Y1);//et la position de référence

            temp=temp/1000+8000;
            temp2=temp2/1000+8000;
       }
       else{
             temp= Float.intBitsToFloat(ip.getPixel(offscreenX,offscreenY));
             temp2= Float.intBitsToFloat(ip.getPixel(X1,Y1));
       }
       temp=temp-temp2;//donc on calcule l'écart entre les 2
       val="dx= "+temp;
       dist=temp*temp; //onmet dans dist l'écart au carré
       ip =IS.getProcessor(3);
       if (color){ //pour la couche Y même chose
           temp= ip.getPixel(offscreenX,offscreenY);
           temp2= ip.getPixel(X1,Y1);

            temp=temp/1000+8000;
            temp2=temp2/1000+8000;
       }
       else{
             temp= Float.intBitsToFloat(ip.getPixel(offscreenX,offscreenY));
             temp2= Float.intBitsToFloat(ip.getPixel(X1,Y1));
       }
       temp=temp-temp2;
       val=val+" dy= "+temp;//sauf que l'écart est stocké à la suite de celui en x
       dist=dist+temp*temp; //et que l'écart au carré est ajouté à dist
       ip =IS.getProcessor(4);
       if (color){ //pour la couche Z même chose
           temp= ip.getPixel(offscreenX,offscreenY);
           temp2= ip.getPixel(X1,Y1);

            temp=temp/1000+8000;
            temp2=temp2/1000+8000;
       }
       else{
             temp= Float.intBitsToFloat(ip.getPixel(offscreenX,offscreenY));
             temp2= Float.intBitsToFloat(ip.getPixel(X1,Y1));
       }
       temp=temp-temp2;
       val=val+" dz= "+temp;
       dist=dist+temp*temp;
       dist= Math.sqrt(dist); //plus qu'à prendre la racine pour avoir la distance entre les points
       val=val+" distance= "+dist;
   }
   IJ.showStatus  (val); //et on affiche val(qui contient soit les coordonnées, soit les écarts)
dans la barre de status d'imageJ


   }


}
```

.6 planmoyen

```
import java.io.*;
import ij.*;
import ij.io.*;
import ij.gui.*;
import ij.plugin.filter.PlugInFilter;
import ij.process.*;
import java.awt.*;
import ij.gui.Roi.*;
import ij.gui.PolygonRoi.*;
public class planmoyen_ implements PlugInFilter {
private static String defaultDirectory = null;
public ImagePlus imp;//l'image solide
public int compteur;
public double[] cx; //les valeurs de x, y, et z pour les pixels selectionnés
public double[] cy;
public double[] cz;
public boolean estligne;
public Polygon selection;

  public int setup(String arg, ImagePlus imp2) {
    estligne=false;
    imp=imp2;
    Roi monroi= imp.getRoi();
    int type = monroi.getType();
    if (type>4&&type<8){//si la selection est de type "ligne"
        estligne=true;
        selection=monroi.getPolygon();
    }
compteur=0;
return DOES_32+DOES_RGB+DOES_STACKS+SUPPORTS_MASKING;


}


  public void run(ImageProcessor ip) {
   if (ip instanceof ColorProcessor){//le codage de la valeur des couches distances est différent en
couleur(24BIT) et en gris(32bit)
      if (estligne==false){ //si la selection n'est pas linéaire
          if(compteur==1){
            planmoyenxc(ip);//on lit les couches x, y et Z
          }
          if(compteur==2){
             planmoyenyc(ip);
          }
          if(compteur==3){
             planmoyenzc(ip);
             boolean points = planmoyen();  //puis on calcul le plan moyen
          }
      }
      else{ //sinon il ne faut lire que les lignes du polygone de selection
          if(compteur==1){
            lignexc(ip);//on lit les couches x, y et Z
          }
          if(compteur==2){
             ligneyc(ip);
          }
          if(compteur==3){
             lignezc(ip);
             boolean points = planmoyen();  //puis on calcul le plan moyen
          }
       }
    }
```



```
    else{
       if (estligne==false){
           if(compteur==1){
             planmoyenx(ip);
           }
           if(compteur==2){
              planmoyeny(ip);
           }
           if(compteur==3){
              planmoyenz(ip);
              boolean points = planmoyen();
           }
       }
       else{ //sinon il ne faut lire que les lignes du polygone de selection
           if(compteur==1){
             lignex(ip);//on lit les couches x, y et Z
           }
           if(compteur==2){
              ligney(ip);
           }
           if(compteur==3){
              lignez(ip);
              boolean points = planmoyen();  //puis on calcul le plan moyen
           }
       }
    }
    compteur=compteur+1;
}
  public void lignexc (ImageProcessor ip) {
    int[] pixels = (int[])ip.getPixels();
    int width = ip.getWidth();
    int height=ip.getHeight();
    int offset, i;
    int xmin=width;
    int xmax=0;
    int ymin=height;
    int ymax=0;
    int k4 = 0;
    for (int j=0; j<selection.npoints; j++) {//on calcule la taille maxi de la selection en nb de pixel
        if(selection.ypoints[j]<ymin){
            ymin=selection.ypoints[j];
        }
        if(selection.ypoints[j]>ymax){
            ymax=selection.ypoints[j];
        }
         if(selection.xpoints[j]<xmin){
            xmin=selection.xpoints[j];
        }
        if(selection.xpoints[j]>xmax){
            xmax=selection.xpoints[j];
        }
    }
    xmax=xmax-xmin;
    ymax=ymax-ymin;
    xmax=xmax*xmin;
    cx = new double[xmax];
    cy = new double[xmax];
    cz = new double[xmax];
    for (int j=0; j<selection.npoints-1; j++){// pour chaque ligne de la polyligne de selection
        ymin=selection.ypoints[j];
        xmin=selection.xpoints[j];
        ymax=selection.ypoints[j+1];
        xmax=selection.xpoints[j+1];
        if (xmin>xmax){
            int temp=xmax;
            xmax=xmin;
            xmin=temp;
            temp=ymax;
```

```
            ymax=ymin;
            ymin=temp;
        }
        float pente=0;
        if(xmax!=xmin){
            pente=(float)(ymax-ymin)/(float)(xmax-xmin);//on calcule la pente de la droite (voir + bas
 si pente infinie)
            for (int x=xmin;x<xmax;x++){//pour tous les x du segment de droites
                int y1=(int)(pente*(x-xmin))+ymin;
                int y2=(int)(pente*(x+1-xmin))+ymin;
                for(int y =Math.min(y1,y2);y<Math.max(y1,y2)+1;y++){//pour tous les y du sgment entre
 x et x+1
                    i = y*width + x;
                    cx[k4] = (double)pixels[i]/1000+8000;//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }

        }
        else{//on a xmin=xmax donc une pente infinie, il faut donc mettre dans notre tableau tous les
 points de (xmin,ymin) à (xmin,ymax)
            ymin=selection.ypoints[j];
            ymax=selection.ypoints[j+1];
            if (ymax>ymin){
                for (int y=ymin;y<ymax;y++){
                    i = y*width + xmin;
                    cx[k4] = (double)pixels[i]/1000+8000;//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
            else{
                for (int y=ymin;y>ymax;y--){
                    i = y*width + xmin;
                    cx[k4] = (double)pixels[i]/1000+8000;//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
        }
    }
    ymin=selection.ypoints[selection.npoints-1];
    xmin=selection.xpoints[selection.npoints-1];
    i = ymin*width + xmin;
    cx[k4] = (double)pixels[i]/1000+8000;



}
  public void ligneyc (ImageProcessor ip) {
    int[] pixels = (int[])ip.getPixels();
    int width = ip.getWidth();
    int offset, i;
    int xmin;
    int xmax;
    int ymin;
    int ymax;
    int k4 = 0;

    for (int j=0; j<selection.npoints-1; j++){// pour chaque ligne de la polyligne de selection
        ymin=selection.ypoints[j];
        xmin=selection.xpoints[j];
        ymax=selection.ypoints[j+1];
        xmax=selection.xpoints[j+1];
        if (xmin>xmax){
            int temp=xmax;
            xmax=xmin;
            xmin=temp;
            temp=ymax;
            ymax=ymin;
```



```
            ymin=temp;
        }
        float pente=0;
        if(xmax!=xmin){
            pente=(float)(ymax-ymin)/(float)(xmax-xmin);//on calcule la pente de la droite (voir + bas
 si pente infinie)
            for (int x=xmin;x<xmax;x++){//pour tous les x du segment de droites
                int y1=(int)(pente*(x-xmin))+ymin;
                int y2=(int)(pente*(x+1-xmin))+ymin;
                for(int y =Math.min(y1,y2);y<Math.max(y1,y2)+1;y++){//pour tous les y du sgment entre
 x et x+1
                    i = y*width + x;
                    cy[k4] = (double)pixels[i]/1000+8000;//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
        }
        else{//on a xmin=xmax donc une pente infinie, il faut donc mettre dans notre tableau tous les
 points de (xmin,ymin) à (xmin,ymax)
            ymin=selection.ypoints[j];
            ymax=selection.ypoints[j+1];
            if (ymax>ymin){
                for (int y=ymin;y<ymax;y++){
                    i = y*width + xmin;
                    cy[k4] = (double)pixels[i]/1000+8000;//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
            else{
                for (int y=ymin;y>ymax;y--){
                    i = y*width + xmin;
                    cy[k4] = (double)pixels[i]/1000+8000;//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
        }
    }
    ymin=selection.ypoints[selection.npoints-1];
    xmin=selection.xpoints[selection.npoints-1];
    i = ymin*width + xmin;
    cy[k4] = (double)pixels[i]/1000+8000;


}
 public void lignezc (ImageProcessor ip) {
    int[] pixels = (int[])ip.getPixels();
    int width = ip.getWidth();
    int offset, i;
    int xmin;
    int xmax;
    int ymin;
    int ymax;
    compteur=0;

    for (int j=0; j<selection.npoints-1; j++){// pour chaque ligne de la polyligne de selection
        ymin=selection.ypoints[j];
        xmin=selection.xpoints[j];
        ymax=selection.ypoints[j+1];
        xmax=selection.xpoints[j+1];
        if (xmin>xmax){
            int temp=xmax;
            xmax=xmin;
            xmin=temp;
            temp=ymax;
            ymax=ymin;
            ymin=temp;
        }
        float pente=0;
```

```
       if(xmax!=xmin){
           pente=(float)(ymax-ymin)/(float)(xmax-xmin);//on calcule la pente de la droite (voir + bas
 si pente infinie)
           for (int x=xmin;x<xmax;x++){//pour tous les x du segment de droites
               int y1=(int)(pente*(x-xmin))+ymin;
               int y2=(int)(pente*(x+1-xmin))+ymin;
               for(int y =Math.min(y1,y2);y<Math.max(y1,y2)+1;y++){//pour tous les y du sgment entre
x et x+1
                   i = y*width + x;
                   cz[compteur] = (double)pixels[i]/1000+8000;//ca nous fait un point dans notre tableau
                   compteur = compteur + 1;
               }
           }

       }
       else{//on a xmin=xmax donc une pente infinie, il faut donc mettre dans notre tableau tous les
points de (xmin,ymin) à (xmin,ymax)
           ymin=selection.ypoints[j];
           ymax=selection.ypoints[j+1];
           if (ymax>ymin){
               for (int y=ymin;y<ymax;y++){
                   i = y*width + xmin;
                   cz[compteur] = (double)pixels[i]/1000+8000;//ca nous fait un point dans notre tableau
                   compteur = compteur + 1;
               }
           }
           else{
               for (int y=ymin;y>ymax;y--){
                   i = y*width + xmin;
                   cz[compteur] = (double)pixels[i]/1000+8000;//ca nous fait un point dans notre tableau
                   compteur = compteur + 1;
               }
           }
       }
   }
   ymin=selection.ypoints[selection.npoints-1];
   xmin=selection.xpoints[selection.npoints-1];
   i = ymin*width + xmin;
   cz[compteur] = (double)pixels[i]/1000+8000;
   compteur++;


}
   public void lignex (ImageProcessor ip) {
   float[] pixels = (float[])ip.getPixels();
   int width = ip.getWidth();
   int height=ip.getHeight();
   int offset, i;
   int xmin=width;
   int xmax=0;
   int ymin=height;
   int ymax=0;
   int k4 = 0;
   for (int j=0; j<selection.npoints; j++) {//on calcule la taille maxi de la selection en nb de pixel
       if(selection.ypoints[j]<ymin){
           ymin=selection.ypoints[j];
       }
       if(selection.ypoints[j]>ymax){
           ymax=selection.ypoints[j];
       }
        if(selection.xpoints[j]<xmin){
           xmin=selection.xpoints[j];
       }
       if(selection.xpoints[j]>xmax){
           xmax=selection.xpoints[j];
       }
   }
   xmax=xmax-xmin;
```



```
   ymax=ymax-ymin;
   xmax=xmax*xmin;
   cx = new double[xmax];
   cy = new double[xmax];
   cz = new double[xmax];
   for (int j=0; j<selection.npoints-1; j++){// pour chaque ligne de la polyligne de selection
       ymin=selection.ypoints[j];
       xmin=selection.xpoints[j];
       ymax=selection.ypoints[j+1];
       xmax=selection.xpoints[j+1];
       if (xmin>xmax){
           int temp=xmax;
           xmax=xmin;
           xmin=temp;
           temp=ymax;
           ymax=ymin;
           ymin=temp;
       }
       float pente=0;
       if(xmax!=xmin){
           pente=(float)(ymax-ymin)/(float)(xmax-xmin);//on calcule la pente de la droite (voir + bas
si pente infinie)
           for (int x=xmin;x<xmax;x++){//pour tous les x du segment de droites
               int y1=(int)(pente*(x-xmin))+ymin;
               int y2=(int)(pente*(x+1-xmin))+ymin;
               for(int y =Math.min(y1,y2);y<Math.max(y1,y2)+1;y++){//pour tous les y du sgment entre
x et x+1
                   i = y*width + x;
                   cx[k4] = pixels[i];//ca nous fait un point dans notre tableau
                   k4 = k4 + 1;
               }
           }

       }
       else{//on a xmin=xmax donc une pente infinie, il faut donc mettre dans notre tableau tous les
points de (xmin,ymin) à (xmin,ymax)
           ymin=selection.ypoints[j];
           ymax=selection.ypoints[j+1];
           if (ymax>ymin){
               for (int y=ymin;y<ymax;y++){
                   i = y*width + xmin;
                   cx[k4] = pixels[i];//ca nous fait un point dans notre tableau
                   k4 = k4 + 1;
               }
           }
           else{
               for (int y=ymin;y>ymax;y--){
                   i = y*width + xmin;
                   cx[k4] = pixels[i];//ca nous fait un point dans notre tableau
                   k4 = k4 + 1;
               }
           }
       }
   }
   ymin=selection.ypoints[selection.npoints-1];
   xmin=selection.xpoints[selection.npoints-1];
   i = ymin*width + xmin;
   cx[k4] = pixels[i];


}
 public void ligney (ImageProcessor ip) {
   float[] pixels = (float[])ip.getPixels();
   int width = ip.getWidth();
   int offset, i;
   int xmin;
   int xmax;
```

```
    int ymin;
    int ymax;
    int k4 = 0;

    for (int j=0; j<selection.npoints-1; j++){// pour chaque ligne de la polyligne de selection
        ymin=selection.ypoints[j];
        xmin=selection.xpoints[j];
        ymax=selection.ypoints[j+1];
        xmax=selection.xpoints[j+1];
        if (xmin>xmax){
            int temp=xmax;
            xmax=xmin;
            xmin=temp;
            temp=ymax;
            ymax=ymin;
            ymin=temp;
        }
        float pente=0;
        if(xmax!=xmin){
            pente=(float)(ymax-ymin)/(float)(xmax-xmin);//on calcule la pente de la droite (voir + bas
 si pente infinie)
            for (int x=xmin;x<xmax;x++){//pour tous les x du segment de droites
                int y1=(int)(pente*(x-xmin))+ymin;
                int y2=(int)(pente*(x+1-xmin))+ymin;
                for(int y =Math.min(y1,y2);y<Math.max(y1,y2)+1;y++){//pour tous les y du sgment entre
x et x+1
                    i = y*width + x;
                    cy[k4] = pixels[i];//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
        }
        else{//on a xmin=xmax donc une pente infinie, il faut donc mettre dans notre tableau tous les
points de (xmin,ymin) à (xmin,ymax)
            ymin=selection.ypoints[j];
            ymax=selection.ypoints[j+1];
            if (ymax>ymin){
                for (int y=ymin;y<ymax;y++){
                    i = y*width + xmin;
                    cy[k4] = pixels[i];//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
            else{
                for (int y=ymin;y>ymax;y--){
                    i = y*width + xmin;
                    cy[k4] = pixels[i];//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
        }
    }
    ymin=selection.ypoints[selection.npoints-1];
    xmin=selection.xpoints[selection.npoints-1];
    i = ymin*width + xmin;
    cy[k4] = pixels[i];


 }
  public void lignez (ImageProcessor ip) {
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
    int offset, i;
    int xmin;
    int xmax;
    int ymin;
    int ymax;
    compteur=0;
```



```
    for (int j=0; j<selection.npoints-1; j++){// pour chaque ligne de la polyligne de selection
        ymin=selection.ypoints[j];
        xmin=selection.xpoints[j];
        ymax=selection.ypoints[j+1];
        xmax=selection.xpoints[j+1];
        if (xmin>xmax){
            int temp=xmax;
            xmax=xmin;
            xmin=temp;
            temp=ymax;
            ymax=ymin;
            ymin=temp;
        }
        float pente=0;
        if(xmax!=xmin){
            pente=(float)(ymax-ymin)/(float)(xmax-xmin);//on calcule la pente de la droite (voir + bas
si pente infinie)
            for (int x=xmin;x<xmax;x++){//pour tous les x du segment de droites
                int y1=(int)(pente*(x-xmin))+ymin;
                int y2=(int)(pente*(x+1-xmin))+ymin;
                for(int y =Math.min(y1,y2);y<Math.max(y1,y2)+1;y++){//pour tous les y du sgment entre
x et x+1
                    i = y*width + x;
                    cz[compteur] = pixels[i];//ca nous fait un point dans notre tableau
                    compteur = compteur + 1;
                }
            }

        }
        else{//on a xmin=xmax donc une pente infinie, il faut donc mettre dans notre tableau tous les
points de (xmin,ymin) à (xmin,ymax)
            ymin=selection.ypoints[j];
            ymax=selection.ypoints[j+1];
            if (ymax>ymin){
                for (int y=ymin;y<ymax;y++){
                    i = y*width + xmin;
                    cz[compteur] = pixels[i];//ca nous fait un point dans notre tableau
                    compteur = compteur + 1;
                }
            }
            else{
                for (int y=ymin;y>ymax;y--){
                    i = y*width + xmin;
                    cz[compteur] = pixels[i];//ca nous fait un point dans notre tableau
                    compteur = compteur + 1;
                }
            }
        }
    }
    ymin=selection.ypoints[selection.npoints-1];
    xmin=selection.xpoints[selection.npoints-1];
    i = ymin*width + xmin;
    cz[compteur] = pixels[i];
    compteur++;
 }


  public void planmoyenxc(ImageProcessor ip) {
    int k4 = 0; //compteur du nombre de points
    byte[] masque = (byte[]) ip.getMaskArray();//les selections non rectangle sont gérés par un rectangle
 + un tableau de bit
    int[] pixels = (int[])ip.getPixels();
    int width = ip.getWidth();
    Rectangle r = ip.getRoi();//le rectangle contenant la selection
int offset, i;
    cx = new double[r.height*r.width];
    cy = new double[r.height*r.width];
```

```
    cz = new double[r.height*r.width];
    if(masque!=null){
        for (int y=r.y; y<(r.y+r.height); y++) {
     offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {//pour tout les points du rectangle contenant la selection
              i = offset + x;
              cx[k4] = ((double)pixels[i]/1000+8000)*(-masque[k4]);//cx[numéro du point] = 0 si le point
n'est pas dans le masque, le x du point si le point est dans le masque
              k4 = k4 + 1;
              //pour les images couleur le pixel est un entier(24BIT= à la valeur de x en mm +8000m,
 (il faut donc -8000<x<8000 voir le plugin xyzreaderc)
              //pour les images gris le pixel est directement un décimal flotant 32bit
        }
       }
     }
     else {//si il n'y a pas de masque, c'est plus simple...
         for (int y=r.y; y<(r.y+r.height); y++) {
       offset = y*width;
      for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cx[k4] = ((double)pixels[i]/1000+8000);
              k4 = k4 + 1;
              }
          }
     }
   }
   public void planmoyenyc(ImageProcessor ip) {//voir planmoyenxc
     int k4 = 0;
     byte[] masque = (byte[]) ip.getMaskArray();
     int[] pixels = (int[])ip.getPixels();
     int width = ip.getWidth();
     Rectangle r = ip.getRoi();
 int offset, i;
 if(masque!=null){
     for (int y=r.y; y<(r.y+r.height); y++) {
      offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cy[k4] = ((double)pixels[i]/1000+8000)*(-masque[k4]);
              k4 = k4 + 1;
              }
          }
       }
       else {
         for (int y=r.y; y<(r.y+r.height); y++) {
      offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cy[k4] = ((double)pixels[i]/1000+8000);
              k4 = k4 + 1;
              }
          }
      }
   }
 public void planmoyenzc(ImageProcessor ip) {//voir planmoyenxc
     compteur = 0;
     byte[] masque = (byte[]) ip.getMaskArray();
     int[] pixels = (int[])ip.getPixels();
     int width = ip.getWidth();
     Rectangle r = ip.getRoi();
 int offset, i;
     if(masque!=null){
         for (int y=r.y; y<(r.y+r.height); y++) {
      offset = y*width;
      for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cz[compteur] = ((double)pixels[i]/1000+8000)*(-masque[compteur]);
              compteur = compteur + 1;
```



```
        }
       }
     }
     else{
        for (int y=r.y; y<(r.y+r.height); y++) {
      offset = y*width;
      for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cz[compteur] = ((double)pixels[i]/1000+8000);
              compteur = compteur + 1;

        }
       }
     }
}


  public void planmoyenx(ImageProcessor ip) {//voir planmoyenxc
    int k4 = 0;
    byte[] masque = (byte[]) ip.getMaskArray();
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
    Rectangle r = ip.getRoi();
int offset, i;
    cx = new double[r.height*r.width];
    cy = new double[r.height*r.width];
    cz = new double[r.height*r.width];
    if(masque!=null){
        for (int y=r.y; y<(r.y+r.height); y++) {
     offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cx[k4] = pixels[i]*(-masque[k4]);
              k4 = k4 + 1;
        }
       }
     }
     else {
        for (int y=r.y; y<(r.y+r.height); y++) {
      offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cx[k4] = pixels[i];
              k4 = k4 + 1;
              }
          }
     }
   }
 public void planmoyeny(ImageProcessor ip) {//voir planmoyenxc
    int k4 = 0;
    byte[] masque = (byte[]) ip.getMaskArray();
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
    Rectangle r = ip.getRoi();
int offset, i;
if(masque!=null){
     for (int y=r.y; y<(r.y+r.height); y++) {
      offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cy[k4] = pixels[i]*(-masque[k4]);
              k4 = k4 + 1;
              }
          }
       }
       else {
         for (int y=r.y; y<(r.y+r.height); y++) {
      offset = y*width;
```

```
    for (int x=r.x; x<(r.x+r.width); x++) {
            i = offset + x;
            cy[k4] = pixels[i];
            k4 = k4 + 1;
            }
        }
     }
  }
public void planmoyenz(ImageProcessor ip) {//voir planmoyenxc
    compteur = 0;
    byte[] masque = (byte[]) ip.getMaskArray();
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
    Rectangle r = ip.getRoi();
int offset, i;
    if(masque!=null){
        for (int y=r.y; y<(r.y+r.height); y++) {
     offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
            i = offset + x;
            cz[compteur] = pixels[i]*(-masque[compteur]);
            compteur = compteur + 1;
       }
      }
    }
   else{
       for (int y=r.y; y<(r.y+r.height); y++) {
     offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
            i = offset + x;
            cz[compteur] = pixels[i];
            compteur = compteur + 1;
       }
      }
    }
  }
}

 public boolean planmoyen( ) {
  double N;
  int dist = 0;
  double a = 0;
  double b = 0;
  double c = 0;
  double d = 0;
  double ka1 = 0;
  double ka2 = 0;
  double ka3 = 0;
  double kb1 = 0;
  double kb2 = 0;
  double kc1 = 0;
  double k1 = 0;
  double k2 = 0;
  double k3 = 0;
  double k4 = 0;
  double k5 = 0;
  double k6 = 0;
  double k7 = 0;
  double k8 = 0;
  double k9 = 0;
  double k10 = 0;
  while(dist<compteur){//résolution exacte du moindre carré pour un plan, compteur est le nombre de
point dans le rectangle de sélection
     if(!(( cx[dist]==0)&&(cy[dist]==0)&&(cy[dist]==0))){//(x,y,z)=(0,0,0)si le point est hors du masque
ou si le pixel ne contient pas de données xyz
         k1 = k1 + (cx[dist] * cx[dist]); //somme des xcarrés
    k2 = k2 + (cy[dist] * cy[dist]); //somme des ycarrés
    k3 = k3 + (cz[dist] * cz[dist]); //somme des zcarrés
         k4 = k4 +1;                      //nb de points
```



```
        k5 = k5 +(cx[dist] * cy[dist]);  //somme des xy
      k6 = k6 +(cx[dist] * cz[dist]);  //somme des xz
    k7 = k7 + cx[dist];              //somme des x
    k8 = k8 +(cy[dist] * cz[dist]);  //somme des yz
    k9 = k9 + cy[dist];              //somme des y
    k10 = k10 + cz[dist];            //somme des z
          }
     dist = dist + 1;//ne sert pas pour la résolution du plan moyen,

  }
     dist=(int)k4;
     if (dist<3)  { //si il y a moins de 3 points, pas de plan
        return false;
     }
     if (k1==0)  {
          showAbout(1 , 0 , 0 , 0 , 0 , dist);//cas k1=0...tous les x=0!
          save( a , b , c , d);
          return true;
     }
     k5 = 2*k5;
     k6 = 2*k6;
     k7 = 2*k7;
     k8 = 2*k8;
     k9 = 2*k9;
     k10 = 2*k10;
     ka1 = k5/k1/2;
     ka2 = k6/k1/2;
     ka3 = k7/k1/2;
     k2 = k2 - (ka1*k5/2);
     k3 = k3 - (ka2*k6/2);
     k4 = k4 - (ka3*k7/2);
     k8 = k8 - (k5*ka2);
     k9 = k9 - (k5*ka3);
     k10 = k10 - (k6*ka3);
     if(k2==0){//alors y/x = cst sur tous les points
      b=1;
      a=ka1*b;
      N=(double) Math.sqrt((a*a)+(b*b));
      b=b/N;
      a=a/N;
        showAbout(a , b , 0 , 0 , 0 , dist);//affichage des coordonnées
        save( a , b , c , d);              //sauvegarde de l'équation
        return true;
     }

     kb1 = k8/k2/2;
     kb2 = k9/k2/2;
     k3 = k3 - (kb1*k8/2);
     k4 = k4 - (kb2*k9/2);
     k10 = k10 - (k8*kb2);
     if(k3==0){//tous les points sont sur un plan passant par (0,0,0)
        c=1;
        b=-kb1;
        a=ka1*kb1-ka2;
        N=(double) Math.sqrt((a*a)+(b*b)+(c*c));
        a=a/N;
        b=b/N;
        c=c/N;
        showAbout(a , b , c , 0 , 0 , dist);
        save( a , b , c , d);
        return true;
     }
     k4 = k4;
     kc1 = (k10/k3)/2;
     k4 = k4 - (kc1*k10/2);
     d = 1;
     c = -kc1;
     b = -kb2 + (kc1*kb1);
```

```
    a = -(ka1*b)+ (ka2*kc1)-ka3;
    N=(double) Math.sqrt((a*a)+(b*b)+(c*c));
    if(N==0){//pas de solution avec d=1->d=0!
       c=1;
       b=-kb1;
       a=ka1*kb1-ka2;
       N=(double) Math.sqrt((a*a)+(b*b)+(c*c));
       a=a/N;
       b=b/N;
       c=c/N;
       k4=k3*k2*k1;
       showAbout(a , b , c , 0 , k4 , dist);
       save( a , b , c , d);
      return true;
    }
    a=a/N;
    b=b/N;
    c=c/N;
    d=d/N;
    k4=d*d*k4;
    k4 = (double)Math.sqrt(k4/dist);//distance quadratique moyenne entre un point et le plan.
    showAbout(a , b , c , d , k4 , dist);//affichage des coordonnées
    save( a , b , c , d);                //sauvegarde de l'équation
    return true;
}
       public void save(double a, double b, double c, double d){
     String direct=saveAsAscii();//répertoire de sauvegarde, etc...
   String coord = " ";
   int j=0;
   if(c<0){//on prend la normale montante
     c=-c;
     b=-b;
     a=-a;
     d=-d;
   }
   double dip = Math.acos(c)* 180 / 3.1415926535;//plongement en degré //attention convention nord
en Y
   double dir = Math.atan(a/b)* 180 / 3.1415926535;//direction de plongement en degré
   if(b<0){//si b<0->on a pas pris la bonne solution de l'arctangente
     dir=dir+180;
   }
   int trace=drawtrace( dip,dir);//dessin du plan sur le stéréogramme
   if(trace<0){
      trace=trace+16777216;
   }
   String bla=direct+"roi"+trace;
   RoiEncoder saveroi = new RoiEncoder(bla);//sauvegarde de la zone de sélection dans forme de ficher.

   try {

   saveroi.write(imp.getRoi());
     OutputStream out = new BufferedOutputStream(new FileOutputStream(direct));//sauvegarde de
l'équation dans un fichier temporaire
     OutputStreamWriter osw = new OutputStreamWriter(out);

                    // (a    b    c    d    plongement    direction
         osw.write(IJ.d2s(a,9)+"\t"+IJ.d2s(b,9)+"\t"+IJ.d2s(c,9)+"\t"+IJ.d2s(d,3)+"\t"+IJ.d2s(dip,1)+"\t"+IJ.d2s(dir,1)+"\n
        j++;


       osw.flush();
       osw.close();
       out.close();
     } catch (IOException e) {
     IJ.error("Error writing Ascii file ");
   }
   }
```



```
public String saveAsAscii() {//renvoi le nom dossier_par_défaut+nom_de_l'image_solide_+plan.txttemp
  String directory = OpenDialog.getDefaultDirectory();
  String name = imp.getTitle();
  name = name+"plan.txttemp";
  return directory+name;
  }

   int drawtrace ( double dip, double dir){//tracé du plan dans le stéréogramme
    int[] wList = WindowManager.getIDList();
     if (wList==null) {
         IJ.noImage();
         return -1;
     }
     int j=-1;
     if(wList.length>1){//le stéréogramme est ouvert en deuxième fichier image.
         j=1;


     }
     if (j==-1){ //si le stéréogramme n'existe pas, il faut d'abord le créer
         int w = 501, h = 501;
     ImageProcessor ip = new ColorProcessor(w, h);//on crée donc un nouveau colorprocessor
     int[] pixels = (int[])ip.getPixels();
         for (int i=0; i<501*501;i++){
               pixels[i]=-1;//on initialise tous les pixels sur blanc
         }
         for (int x = 0; x < 251; x++) { //on trace le cercle de rayon 250 pixel
              int y = (int) (Math.sqrt((500-x)*x));
              int i = (250-y)*501+x;
              pixels[i]=0;
              i = (250+y)*501+x;
              pixels[i]=0;
              i = (250-y)*501+500-x;
              pixels[i]=0;
              i = (250+y)*501+500-x;
              pixels[i]=0;
         }
         for (int y = 0; y < 251; y++) {//on trace le cercle de rayon 250 pixel
              int x = (int) (Math.sqrt((500-y)*y));
              int i = (250-x)+y*501;
              pixels[i]=0;
              i = (250+x)+y*501;
              pixels[i]=0;
              i = (250-x)+(500-y)*501;
              pixels[i]=0;
              i = (250+x)+(500-y)*501;
              pixels[i]=0;
         }
       new ImagePlus("stereogramme.tif", ip).show();//on crée l'image du stéréogramme et on l'affiche
       j=1;
        }


       ImagePlus imp2 = WindowManager.getImage(wList[j]);
       ImageProcessor ip2 = imp2.getProcessor();
       int color=ip2.getPixel(0,0);//le pixel (0,0) nous sert de compteur, sa valeur augmente de 1
à chaque nouveau plan tracé
       color++;
       double rayon = Math.tan(3.1415926535*dip/360)*250;
       dir = (dir)*3.1415926535/180;
       for(int x = (int)(248 - rayon * Math.sin (dir));x<(int)(252 - rayon * Math.sin (dir));x++) {
           for (int y = (int)(248 + rayon * Math.cos (dir));y<(int)(252 + rayon * Math.cos (dir));y++){
               ip2.putPixel(x,y,color);//tracé du pôle (dans la couleur dont la valeur correspond au
```

```
numéro de ce plan)
            }
        }
        if(rayon<250){//si le rayon vaut juste 250 (dip=90 ) la trace est une droite

            double rayon2 = 62500/(250-rayon);//calcul du rayon
            //mettre la trace maintenant!
            int xc=(int)(250 - Math.sqrt(rayon2*rayon2-62500) * Math.sin (dir));//calcul du centre
            int yc=(int)(250 + Math.sqrt(rayon2*rayon2-62500)* Math.cos (dir));
            for (int x = 0; x < 502; x++) {//tracé du cercle


                    double temp=(rayon2*rayon2-(x-xc)*(x-xc));
                    if(temp>-1){
                          int y = (int) (Math.sqrt(temp));
                          int i = (yc-y);
                          if (((250-x)*(250-x)+(250-i)*(250-i))<62500){//one ne trace que si c'est à
l'intérieur du contour du stéréogramme
                              ip2.putPixel(x,i,color);
                          }
                          i=yc+y;
                          if(((250-x)*(250-x)+(250-i)*(250-i))<62500){
                              ip2.putPixel(x,i,color);
                          }
                    }
             }
             for (int y = 0; y < 502; y++) {//tracé du cercle (1 y pour tout x et 1 x pour tout y pour
avoir un trait continu)
                    double temp=(rayon2*rayon2-(y-yc)*(y-yc));
                    if(temp>-1){
                        int x = (int) (Math.sqrt(temp));
                        int i = xc-x;
                         if (((250-y)*(250-y)+(250-i)*(250-i))<62500){
                            ip2.putPixel(i,y,color);
                         }
                         i = xc+x;
                         if (((250-y)*(250-y)+(250-i)*(250-i))<62500){
                            ip2.putPixel(i,y,color);
                         }
                    }
                }
            }
        }
        else{//si le rayon vaut juste 250 (dip=90 ) la trace est une droite

            int xa=(int)(250 - 250 * Math.cos (dir));
            int ya=(int)(250 + 250* Math.sin (dir));
            int xb=(int)(250 + 250 * Math.cos (dir));
            int yb=(int)(250 - 250* Math.sin (dir));
            float pente=0;
            if(xb!=xa){//on trace la droite reliant xa,ya à xb,yb
                pente=(float)(yb-ya)/(float)(xb-xa);
                for (int x=Math.min(xa,xb);x<Math.max(xa,xb);x++){
                  int y=(int)(pente*(x-xa))+ya;
                }
                if (pente!=0){
                    pente =1/pente;
                }
            }
            else{
                pente=0;
            }
            for (int y=Math.min(ya,yb);y<Math.max(ya,yb);y++){//dans les deux sens pour avoir un trait
continu
                int x=(int)(pente*(y-ya))+xa;
            }

        }
        ip2.putPixel(0,0,color);//on change la couleur de notre compteur
```



```
        imp2.updateAndDraw();//on affiche l'image que l'on vient de calculer
        return color;
    }



void showAbout( double a, double b, double c, double d, double k4, double dist ) {
    if(c<0){
      c=-c;
      b=-b;
      a=-a;
      d=-d;
    }
    double dip = Math.acos(c)* 180 / 3.1415926535;
    double dir = Math.atan(a/b)* 180 / 3.1415926535;
    if(b<0){
      dir=dir+180;
    }

    IJ.showMessage("plan moyen_...",//afichage du plan moyen avec distance quadratique moyenne et nombre
 de point entrant dans le calcul
"l'équation du plan moyen est : \n" +a+"X + "+b+"Y + "+c+"Z + "+d+" =0\n"+
     "plongement ="+dip+" direction de plongement = "+dir+"\n"+
"la distance quadratique moyenne entre un point et ce plan est" +k4+"\n"+
dist+" points sont présents dans ce plan."
);
}
}
```

.7 surface

```
import ij.gui.Roi.*;
import ij.gui.PolygonRoi.*;
import java.io.*;
import ij.*;
import ij.io.*;
import ij.gui.*;
import ij.plugin.filter.PlugInFilter;
import ij.process.*;
import java.awt.*;

public class surface_ implements PlugInFilter {

private static String defaultDirectory = null;
public ImagePlus imp;
private int compteur;
public Polygon facette;
public float[] fx;
public float[] fy;
public float[] fz;
  public int setup(String arg, ImagePlus imp) {

//on commence par mettre de coté le polygone de selection
compteur=0;
Roi monroi= imp.getRoi();
int type = monroi.getType();
    if(type<4)  {
        facette = monroi.getPolygon();
        return DOES_32+DOES_STACKS+SUPPORTS_MASKING;
    }

    IJ.showMessage("type...",
type+"n'est pas un polygone!!!");
    return DOES_32;

}


  public void run(ImageProcessor ip) {

   if(compteur==1){
        facettex(ip);
   }
   if(compteur==2){
         facettey(ip);
 }
   if(compteur==3){
         facettez(ip);

        boolean aire = surface();

 }
 compteur=compteur+1;
}



public void facettex(ImageProcessor ip) {
    int k4 = 0;
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
```



```
    int offset, i;
    fx = new float[facette.npoints];
    fy = new float[facette.npoints];
    fz = new float[facette.npoints];
    for (int j=0; j<facette.npoints; j++) {
       i = facette.ypoints[j]*width + facette.xpoints[j];
       fx[j] = pixels[i];
    }
  }

public void facettey(ImageProcessor ip) {
    int k4 = 0;
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
    int offset, i;
    for (int j=0; j<facette.npoints; j++) {
       i = facette.ypoints[j]*width + facette.xpoints[j];
       fy[j] = pixels[i];
    }
  }

public void facettez(ImageProcessor ip) {
    int k4 = 0;
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
    int offset, i;
    for (int j=0; j<facette.npoints; j++) {
       i = facette.ypoints[j]*width + facette.xpoints[j];
       fz[j] = pixels[i];
    }
  }

public boolean surface(){
    float x=0;
    float y=0;
    float z=0;
    float xa=0;
    float ya=0;
    float za=0;
    float xb=0;
    float yb=0;
    float zb=0;

    for (int j=2; j<facette.npoints; j++){
         xa=fx[j]-fx[j-1];
         xb=fx[j]-fx[0];
         ya=fy[j]-fy[j-1];
         yb=fy[j]-fy[0];
         za=fz[j]-fz[j-1];
         zb=fz[j]-fz[0];
         x=x+ya*zb-yb*za;
         y=y+za*xb-zb*xa;
         z=z+xa*yb-xb*ya;
    }
    float N =(float) Math.sqrt(x*x+y*y+z*z);
    showAbout(N);
    return true;
}
```

```
void showAbout( float N ) {


    IJ.showMessage("surface_...",
"la surface mesurée est de "+N+"m2"

);
}
}
```


.8 listing

```
import java.io.*;
import ij.*;
import ij.io.*;
import ij.plugin.filter.PlugInFilter;
import ij.process.*;
import java.awt.*;
import ij.gui.Roi.*;
import ij.gui.PolygonRoi.*;
import ij.gui.*;
//enregistre tous les points selectionnés dans un fichier
public class listing_ implements PlugInFilter {

private static String defaultDirectory = null;
public ImagePlus imp;
private int compteur;
public float[] cx;
public float[] cy;
public float[] cz;
public boolean estligne;
public Polygon selection;

  public int setup(String arg, ImagePlus imp2) {
    estligne=false;
    imp=imp2;
    Roi monroi= imp.getRoi();
    int type = monroi.getType();
    if (type>4&&type<8){//si la selection est de type "ligne"
        estligne=true;
        selection=monroi.getPolygon();
    }

compteur=0;
return DOES_32+DOES_RGB+DOES_STACKS+SUPPORTS_MASKING;


}


  public void run(ImageProcessor ip) {//voir le plugin planmoyen pour la lecture des coordonnées
    if (estligne){
       if (ip instanceof ColorProcessor){

          if(compteur==1){
            lignexc(ip);
          }
          if(compteur==2){
             ligneyc(ip);
          }
          if(compteur==3){
             lignezc(ip);
             save();
          }
       }
       else{
          if(compteur==1){
            lignex(ip);
          }
          if(compteur==2){
             ligney(ip);
          }
          if(compteur==3){
             lignez(ip);
             save();
```

```
        }
      }
    }
    else{
      if (ip instanceof ColorProcessor){

          if(compteur==1){
            planmoyenxc(ip);
          }
          if(compteur==2){
             planmoyenyc(ip);
          }
          if(compteur==3){
             planmoyenzc(ip);
             save();
          }
        }
        else{
          if(compteur==1){
            planmoyenx(ip);
          }
          if(compteur==2){
             planmoyeny(ip);
          }
          if(compteur==3){
             planmoyenz(ip);
             save();
          }
        }
    }

    compteur=compteur+1;
  }
  public void lignexc (ImageProcessor ip) {
    int[] pixels = (int[])ip.getPixels();
    int width = ip.getWidth();
    int height=ip.getHeight();
    int offset, i;
    int xmin=width;
    int xmax=0;
    int ymin=height;
    int ymax=0;
    int k4 = 0;
    for (int j=0; j<selection.npoints; j++) {//on calcule la taille maxi de la selection en nb de pixel
        if(selection.ypoints[j]<ymin){
            ymin=selection.ypoints[j];
        }
        if(selection.ypoints[j]>ymax){
            ymax=selection.ypoints[j];
        }
         if(selection.xpoints[j]<xmin){
            xmin=selection.xpoints[j];
        }
        if(selection.xpoints[j]>xmax){
            xmax=selection.xpoints[j];
        }
    }
    xmax=xmax-xmin;
    ymax=ymax-ymin;
    xmax=xmax*xmin;
    cx = new float[xmax];
    cy = new float[xmax];
    cz = new float[xmax];
    for (int j=0; j<selection.npoints-1; j++){// pour chaque ligne de la polyligne de selection
        ymin=selection.ypoints[j];
        xmin=selection.xpoints[j];
        ymax=selection.ypoints[j+1];
        xmax=selection.xpoints[j+1];
```



```
        if (xmin>xmax){
            int temp=xmax;
            xmax=xmin;
            xmin=temp;
            temp=ymax;
            ymax=ymin;
            ymin=temp;
        }
        float pente=0;
        if(xmax!=xmin){
            pente=(float)(ymax-ymin)/(float)(xmax-xmin);//on calcule la pente de la droite (voir + bas
si pente infinie)
            for (int x=xmin;x<xmax;x++){//pour tous les x du segment de droites
                int y1=(int)(pente*(x-xmin))+ymin;
                int y2=(int)(pente*(x+1-xmin))+ymin;
                for(int y =Math.min(y1,y2);y<Math.max(y1,y2)+1;y++){//pour tous les y du sgment entre
x et x+1
                    i = y*width + x;
                    cx[k4] = ((float)pixels[i])/1000+8000;//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }

        }
        else{//on a xmin=xmax donc une pente infinie, il faut donc mettre dans notre tableau tous les
points de (xmin,ymin) à (xmin,ymax)
            ymin=selection.ypoints[j];
            ymax=selection.ypoints[j+1];
            if (ymax>ymin){
                for (int y=ymin;y<ymax;y++){
                    i = y*width + xmin;
                    cx[k4] = ((float)pixels[i])/1000+8000;//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
            else{
                for (int y=ymin;y>ymax;y--){
                    i = y*width + xmin;
                    cx[k4] = ((float)pixels[i])/1000+8000;//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
        }
    }
    ymin=selection.ypoints[selection.npoints-1];
    xmin=selection.xpoints[selection.npoints-1];
    i = ymin*width + xmin;
    cx[k4] = ((float)pixels[i])/1000+8000;


}
  public void ligneyc (ImageProcessor ip) {
    int[] pixels = (int[])ip.getPixels();
    int width = ip.getWidth();
    int offset, i;
    int xmin;
    int xmax;
    int ymin;
    int ymax;
    int k4 = 0;

    for (int j=0; j<selection.npoints-1; j++){// pour chaque ligne de la polyligne de selection
        ymin=selection.ypoints[j];
        xmin=selection.xpoints[j];
        ymax=selection.ypoints[j+1];
        xmax=selection.xpoints[j+1];
        if (xmin>xmax){
```

```
            int temp=xmax;
            xmax=xmin;
            xmin=temp;
            temp=ymax;
            ymax=ymin;
            ymin=temp;
        }
        float pente=0;
        if(xmax!=xmin){
            pente=(float)(ymax-ymin)/(float)(xmax-xmin);//on calcule la pente de la droite (voir + bas
si pente infinie)
            for (int x=xmin;x<xmax;x++){//pour tous les x du segment de droites
                int y1=(int)(pente*(x-xmin))+ymin;
                int y2=(int)(pente*(x+1-xmin))+ymin;
                for(int y =Math.min(y1,y2);y<Math.max(y1,y2)+1;y++){//pour tous les y du sgment entre
x et x+1
                    i = y*width + x;
                    cy[k4] = ((float)pixels[i])/1000+8000;//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
        }
        else{//on a xmin=xmax donc une pente infinie, il faut donc mettre dans notre tableau tous les
points de (xmin,ymin) à (xmin,ymax)
            ymin=selection.ypoints[j];
            ymax=selection.ypoints[j+1];
            if (ymax>ymin){
                for (int y=ymin;y<ymax;y++){
                    i = y*width + xmin;
                    cy[k4] = ((float)pixels[i])/1000+8000;//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
            else{
                for (int y=ymin;y>ymax;y--){
                    i = y*width + xmin;
                    cy[k4] = ((float)pixels[i])/1000+8000;//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
        }
    }
    ymin=selection.ypoints[selection.npoints-1];
    xmin=selection.xpoints[selection.npoints-1];
    i = ymin*width + xmin;
    cy[k4] = ((float)pixels[i])/1000+8000;


}
 public void lignezc (ImageProcessor ip) {
    int[] pixels = (int[])ip.getPixels();
    int width = ip.getWidth();
    int offset, i;
    int xmin;
    int xmax;
    int ymin;
    int ymax;
    compteur=0;

    for (int j=0; j<selection.npoints-1; j++){// pour chaque ligne de la polyligne de selection
        ymin=selection.ypoints[j];
        xmin=selection.xpoints[j];
        ymax=selection.ypoints[j+1];
        xmax=selection.xpoints[j+1];
        if (xmin>xmax){
            int temp=xmax;
            xmax=xmin;
            xmin=temp;
```



```
            temp=ymax;
            ymax=ymin;
            ymin=temp;
        }
        float pente=0;
        if(xmax!=xmin){
            pente=(float)(ymax-ymin)/(float)(xmax-xmin);//on calcule la pente de la droite (voir + bas
si pente infinie)
            for (int x=xmin;x<xmax;x++){//pour tous les x du segment de droites
                int y1=(int)(pente*(x-xmin))+ymin;
                int y2=(int)(pente*(x+1-xmin))+ymin;
                for(int y =Math.min(y1,y2);y<Math.max(y1,y2)+1;y++){//pour tous les y du sgment entre
x et x+1
                    i = y*width + x;
                    cz[compteur] = ((float)pixels[i])/1000+8000;//ca nous fait un point dans notre tableau
                    compteur = compteur + 1;
                }
            }

        }
        else{//on a xmin=xmax donc une pente infinie, il faut donc mettre dans notre tableau tous les
points de (xmin,ymin) à (xmin,ymax)
            ymin=selection.ypoints[j];
            ymax=selection.ypoints[j+1];
            if (ymax>ymin){
                for (int y=ymin;y<ymax;y++){
                    i = y*width + xmin;
                    cz[compteur] = ((float)pixels[i])/1000+8000;//ca nous fait un point dans notre tableau
                    compteur = compteur + 1;
                }
            }
            else{
                for (int y=ymin;y>ymax;y--){
                    i = y*width + xmin;
                    cz[compteur] = ((float)pixels[i])/1000+8000;//ca nous fait un point dans notre tableau
                    compteur = compteur + 1;
                }
            }
        }
    }
    ymin=selection.ypoints[selection.npoints-1];
    xmin=selection.xpoints[selection.npoints-1];
    i = ymin*width + xmin;
    cz[compteur] = ((float)pixels[i])/1000+8000;
    compteur++;


}
    public void lignex (ImageProcessor ip) {
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
    int height=ip.getHeight();
    int offset, i;
    int xmin=width;
    int xmax=0;
    int ymin=height;
    int ymax=0;
    int k4 = 0;
    for (int j=0; j<selection.npoints; j++) {//on calcule la taille maxi de la selection en nb de pixel
        if(selection.ypoints[j]<ymin){
            ymin=selection.ypoints[j];
        }
        if(selection.ypoints[j]>ymax){
            ymax=selection.ypoints[j];
        }
         if(selection.xpoints[j]<xmin){
            xmin=selection.xpoints[j];
        }
```

```
        if(selection.xpoints[j]>xmax){
            xmax=selection.xpoints[j];
        }
    }
    xmax=xmax-xmin;
    ymax=ymax-ymin;
    xmax=xmax*xmin;
    cx = new float[xmax];
    cy = new float[xmax];
    cz = new float[xmax];
    for (int j=0; j<selection.npoints-1; j++){// pour chaque ligne de la polyligne de selection
        ymin=selection.ypoints[j];
        xmin=selection.xpoints[j];
        ymax=selection.ypoints[j+1];
        xmax=selection.xpoints[j+1];
        if (xmin>xmax){
            int temp=xmax;
            xmax=xmin;
            xmin=temp;
            temp=ymax;
            ymax=ymin;
            ymin=temp;
        }
        float pente=0;
        if(xmax!=xmin){
            pente=(float)(ymax-ymin)/(float)(xmax-xmin);//on calcule la pente de la droite (voir + bas
si pente infinie)
            for (int x=xmin;x<xmax;x++){//pour tous les x du segment de droites
                int y1=(int)(pente*(x-xmin))+ymin;
                int y2=(int)(pente*(x+1-xmin))+ymin;
                for(int y =Math.min(y1,y2);y<Math.max(y1,y2)+1;y++){//pour tous les y du sgment entre
x et x+1
                    i = y*width + x;
                    cx[k4] = pixels[i];//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }

        }
        else{//on a xmin=xmax donc une pente infinie, il faut donc mettre dans notre tableau tous les
points de (xmin,ymin) à (xmin,ymax)
            ymin=selection.ypoints[j];
            ymax=selection.ypoints[j+1];
            if (ymax>ymin){
                for (int y=ymin;y<ymax;y++){
                    i = y*width + xmin;
                    cx[k4] = pixels[i];//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
            else{
                for (int y=ymin;y>ymax;y--){
                    i = y*width + xmin;
                    cx[k4] = pixels[i];//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
        }
    }
    ymin=selection.ypoints[selection.npoints-1];
    xmin=selection.xpoints[selection.npoints-1];
    i = ymin*width + xmin;
    cx[k4] = pixels[i];


}
  public void ligney (ImageProcessor ip) {
```



```
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
    int offset, i;
    int xmin;
    int xmax;
    int ymin;
    int ymax;
    int k4 = 0;

    for (int j=0; j<selection.npoints-1; j++){// pour chaque ligne de la polyligne de selection
        ymin=selection.ypoints[j];
        xmin=selection.xpoints[j];
        ymax=selection.ypoints[j+1];
        xmax=selection.xpoints[j+1];
        if (xmin>xmax){
            int temp=xmax;
            xmax=xmin;
            xmin=temp;
            temp=ymax;
            ymax=ymin;
            ymin=temp;
        }
        float pente=0;
        if(xmax!=xmin){
            pente=(float)(ymax-ymin)/(float)(xmax-xmin);//on calcule la pente de la droite (voir + bas
si pente infinie)
            for (int x=xmin;x<xmax;x++){//pour tous les x du segment de droites
                int y1=(int)(pente*(x-xmin))+ymin;
                int y2=(int)(pente*(x+1-xmin))+ymin;
                for(int y =Math.min(y1,y2);y<Math.max(y1,y2)+1;y++){//pour tous les y du sgment entre
x et x+1
                    i = y*width + x;
                    cy[k4] = pixels[i];//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
        }
        else{//on a xmin=xmax donc une pente infinie, il faut donc mettre dans notre tableau tous les
points de (xmin,ymin) à (xmin,ymax)
            ymin=selection.ypoints[j];
            ymax=selection.ypoints[j+1];
            if (ymax>ymin){
                for (int y=ymin;y<ymax;y++){
                    i = y*width + xmin;
                    cy[k4] = pixels[i];//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
            else{
                for (int y=ymin;y>ymax;y--){
                    i = y*width + xmin;
                    cy[k4] = pixels[i];//ca nous fait un point dans notre tableau
                    k4 = k4 + 1;
                }
            }
        }
    }
    ymin=selection.ypoints[selection.npoints-1];
    xmin=selection.xpoints[selection.npoints-1];
    i = ymin*width + xmin;
    cy[k4] = pixels[i];


}
 public void lignez (ImageProcessor ip) {
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
    int offset, i;
```

```
  int xmin;
  int xmax;
  int ymin;
  int ymax;
  compteur=0;

  for (int j=0; j<selection.npoints-1; j++){// pour chaque ligne de la polyligne de selection
      ymin=selection.ypoints[j];
      xmin=selection.xpoints[j];
      ymax=selection.ypoints[j+1];
      xmax=selection.xpoints[j+1];
      if (xmin>xmax){
          int temp=xmax;
          xmax=xmin;
          xmin=temp;
          temp=ymax;
          ymax=ymin;
          ymin=temp;
      }
      float pente=0;
      if(xmax!=xmin){
          pente=(float)(ymax-ymin)/(float)(xmax-xmin);//on calcule la pente de la droite (voir + bas
si pente infinie)
          for (int x=xmin;x<xmax;x++){//pour tous les x du segment de droites
              int y1=(int)(pente*(x-xmin))+ymin;
              int y2=(int)(pente*(x+1-xmin))+ymin;
              for(int y =Math.min(y1,y2);y<Math.max(y1,y2)+1;y++){//pour tous les y du sgment entre
x et x+1
                  i = y*width + x;
                  cz[compteur] = pixels[i];//ca nous fait un point dans notre tableau
                  compteur = compteur + 1;
              }
          }

      }
      else{//on a xmin=xmax donc une pente infinie, il faut donc mettre dans notre tableau tous les
points de (xmin,ymin) à (xmin,ymax)
          ymin=selection.ypoints[j];
          ymax=selection.ypoints[j+1];
          if (ymax>ymin){
              for (int y=ymin;y<ymax;y++){
                  i = y*width + xmin;
                  cz[compteur] = pixels[i];//ca nous fait un point dans notre tableau
                  compteur = compteur + 1;
              }
          }
          else{
              for (int y=ymin;y>ymax;y--){
                  i = y*width + xmin;
                  cz[compteur] = pixels[i];//ca nous fait un point dans notre tableau
                  compteur = compteur + 1;
              }
          }
      }
  }
  ymin=selection.ypoints[selection.npoints-1];
  xmin=selection.xpoints[selection.npoints-1];
  i = ymin*width + xmin;
  cz[compteur] = pixels[i];
  compteur++;
}


  public void planmoyenxc(ImageProcessor ip) { //voir le plugin planmoyen pour la lecture des coordonnées
    int k4 = 0;
    byte[] masque = (byte[]) ip.getMaskArray();
    int[] pixels = (int[])ip.getPixels();
    int width = ip.getWidth();
```



```
    Rectangle r = ip.getRoi();
int offset, i;
    cx = new float[r.height*r.width];
    cy = new float[r.height*r.width];
    cz = new float[r.height*r.width];
    if(masque!=null){
        for (int y=r.y; y<(r.y+r.height); y++) {
     offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cx[k4] = (float)((float)pixels[i]/1000+8000)*(-masque[k4]);
              k4 = k4 + 1;
        }
      }
    }
    else {
        for (int y=r.y; y<(r.y+r.height); y++) {
      offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cx[k4] =(float) ((float)pixels[i]/1000+8000);
              k4 = k4 + 1;
              }
          }
    }
  }
  public void planmoyenyc(ImageProcessor ip) {//voir le plugin planmoyen pour la lecture des coordonnées
    int k4 = 0;
    byte[] masque = (byte[]) ip.getMaskArray();
    int[] pixels = (int[])ip.getPixels();
    int width = ip.getWidth();
    Rectangle r = ip.getRoi();
int offset, i;
if(masque!=null){
     for (int y=r.y; y<(r.y+r.height); y++) {
      offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cy[k4] =(float) ((float)pixels[i]/1000+8C00)*(-masque[k4]);
              k4 = k4 + 1;
              }
          }
        }
       else {
         for (int y=r.y; y<(r.y+r.height); y++) {
       offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cy[k4] = (float)((float)pixels[i]/1000+8000);
              k4 = k4 + 1;
              }
          }
       }
  }
public void planmoyenzc(ImageProcessor ip) { //voir le plugin planmoyen pour la lecture des coordonnées
    compteur = 0;
    byte[] masque = (byte[]) ip.getMaskArray();
    int[] pixels = (int[])ip.getPixels();
    int width = ip.getWidth();
    Rectangle r = ip.getRoi();
int offset, i;
    if(masque!=null){
        for (int y=r.y; y<(r.y+r.height); y++) {
     offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cz[compteur] = (float)((float)pixels[i]/1000+8000)*(-masque[compteur]);
              compteur = compteur + 1;
```

```
      }
     }
    }
   else{
       for (int y=r.y; y<(r.y+r.height); y++) {
     offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cz[compteur] =(float) ((float)pixels[i]/1000+8000);
              compteur = compteur + 1;

       }
     }
    }
}
```

```
  public void planmoyenx(ImageProcessor ip) {//voir le plugin planmoyen pour la lecture des coordonnées
    int k4 = 0;
    byte[] masque = (byte[]) ip.getMaskArray();
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
    Rectangle r = ip.getRoi();
int offset, i;
    cx = new float[r.height*r.width];
    cy = new float[r.height*r.width];
    cz = new float[r.height*r.width];
    if(masque!=null){
        for (int y=r.y; y<(r.y+r.height); y++) {
     offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cx[k4] = pixels[i]*(-masque[k4]);
              k4 = k4 + 1;
       }
     }
    }
    else {
        for (int y=r.y; y<(r.y+r.height); y++) {
      offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cx[k4] = pixels[i];
              k4 = k4 + 1;
              }
        }
    }
  }
  public void planmoyeny(ImageProcessor ip) {//voir le plugin planmoyen pour la lecture des coordonnées
    int k4 = 0;
    byte[] masque = (byte[]) ip.getMaskArray();
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
    Rectangle r = ip.getRoi();
int offset, i;
if(masque!=null){
     for (int y=r.y; y<(r.y+r.height); y++) {
      offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cy[k4] = pixels[i]*(-masque[k4]);
              k4 = k4 + 1;
              }
         }
        }
```



```
     else {
       for (int y=r.y; y<(r.y+r.height); y++) {
      offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cy[k4] = pixels[i];
              k4 = k4 + 1;
              }
         }
      }
  }
public void planmoyenz(ImageProcessor ip) {//voir le plugin planmoyen pour la lecture des coordonnées
    compteur = 0;
    byte[] masque = (byte[]) ip.getMaskArray();
    float[] pixels = (float[])ip.getPixels();
    int width = ip.getWidth();
    Rectangle r = ip.getRoi();
int offset, i;
    if(masque!=null){
        for (int y=r.y; y<(r.y+r.height); y++) {
     offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cz[compteur] = pixels[i]*(-masque[compteur]);
              compteur = compteur + 1;
       }
      }
    }
   else{
       for (int y=r.y; y<(r.y+r.height); y++) {
     offset = y*width;
     for (int x=r.x; x<(r.x+r.width); x++) {
              i = offset + x;
              cz[compteur] = pixels[i];
              compteur = compteur + 1;
       }
      }
    }
}
```

```
  public void save(){
    String direct=saveAsAscii();
    String coord = " ";
    int j=0;
    try {

      OutputStream out = new BufferedOutputStream(new FileOutputStream(direct));
      OutputStreamWriter osw = new OutputStreamWriter(out);

      for(int i=0; i<compteur; i++){//pour tous les points du rectangle de selection
          if(!(( cx[i]==0)&&(cy[i]==0)&&(cy[i]==0))){//si ils ne sont pas dans le masque, et que les
 coordonnés sont bien présente
           osw.write("point "+ cx[i]+","+cy[i]+","+cz[i]+"\n" );//on écrit ses coordonnées dans le fichier
          j++;
          }
      }
          osw.flush();
          osw.close();
          out.close();
          showAbout(j , compteur);
      } catch (IOException e) {
           IJ.error("Error writing Ascii file ");
      }
}
```

```
 public String saveAsAscii() {
SaveDialog sd = new SaveDialog("Save as ASCII Format", "bla", ".txt");
String directory = sd.getDirectory();
String name = sd.getFileName();
return directory+name;
}


void showAbout( int j,int k ) {
IJ.showMessage("listing_...",
j+"points ont été sauvegardés\n"+
"sur un total de "+k+"pixels" );
}
}
```


.9 trouveintersect

```
import java.io.*;
import ij.*;
import ij.io.*;
import ij.gui.*;
import ij.plugin.PlugIn;
import ij.process.*;
import java.awt.*;
public class trouveintersect_ implements PlugIn {

private static String defaultDirectory = null;
public ImageStack img1; //l'image solide sur laquelle on va chercher les intersection
public ImagePlus sliceimg1;
double a;//les paramètres de l'équation du plan d'intersection
double b;
double c;
double d;
double dist;//la distance maxi entre le plan et un pixel "intersectant"


  public void run(String arg) {
        if (showDialog())
            testintersect();

  }

  public boolean showDialog() {
        int[] wList = WindowManager.getIDList();
        if (wList==null) {
            IJ.noImage();//si aucune image n'est ouverte, on ne fait rien.
            return false;
        }
        String[] titles = new String[wList.length];
        for (int i=0; i<wList.length; i++) {
            ImagePlus imp = WindowManager.getImage(wList[i]);
            if (imp!=null)
                titles[i] = imp.getTitle();
            else
                titles[i] = "";
        }
        GenericDialog gd = new GenericDialog("Plan d'intersection");//sinon on ouvre une boite de
dialogue
        gd.addChoice("Image: ", titles, titles[0]);//pour choisir l'image sur laquelle on teste
l'intersection
        gd.addNumericField("a", 0, 9);//et donner les parammêtres du plan d'intersection
        gd.addNumericField("b", 0, 9);
        gd.addNumericField("c", 0, 9);
        gd.addNumericField("d", 0, 9);
        gd.addNumericField("dist", 0, 9);
        gd.showDialog();
        if (gd.wasCanceled())
            return false;//si la boite de dialogue est annulé on ne fait rien
        int index1 = gd.getNextChoiceIndex();//sinon on récupère les valeurs de la boite de
dialogue
        a=gd.getNextNumber();
        b=gd.getNextNumber();
        c=gd.getNextNumber();
        d=gd.getNextNumber();
        dist=gd.getNextNumber();

        String title1 = titles[index1];
        sliceimg1 = WindowManager.getImage(wList[index1]);
        img1 = sliceimg1.getStack();
        return true;//et on test les intersections.
  }
```

```
public void testintersect() {
   int largeur=img1.getWidth();
   int hauteur=img1.getHeight();
   float x=0;
   float y=0;
   float z=0;
   ImageProcessor ip = img1.getProcessor(1);
   ImageProcessor ip2 = ip.duplicate();//copie de la couche photo de l'image solide pour y mettre
les pixels d'intersection
   for (int x0=0;x0<largeur;x0++){
        for (int y0=0;y0<hauteur;y0++){//pour chaque pixel de l'image solide
            ip = img1.getProcessor(2);//couche "x"
            if (ip instanceof ColorProcessor){//la transformation bits->valeur de x est differente
en image couleur(24BIT) ou gris(32BIT)
                x=((float)ip.getPixel(x0,y0)/1000+8000);
                ip = img1.getProcessor(3);//couche y
                y=((float)ip.getPixel(x0,y0)/1000+8000);
                ip = img1.getProcessor(4);//couche z
                z=((float)ip.getPixel(x0,y0)/1000+8000);
            }

            else{
                x=(Float.intBitsToFloat(ip.getPixel(x0,y0)));
                ip = img1.getProcessor(3);//couche y
                y=(Float.intBitsToFloat(ip.getPixel(x0,y0)));
                ip = img1.getProcessor(4);//couche Z
                z=(Float.intBitsToFloat(ip.getPixel(x0,y0)));
            }
            if(x!=0||y!=0||z!=0){//le code (0,0,0 correspond à pas de données!
                double distance=a*x+b*y+c*z+d;//calcul de la distance entre le pixel et le plan.
                if(distance<0){
                    distance=-distance;//distance prise en valeur absolue
                }
                if(distance<dist){//si le point est suffisament proche du plan

                    ip2.putPixel(x0,y0,-1);//on colorie en blanc le pixel sur la copie ip2
                }
            }


        }
   }
   new ImagePlus("intersection", ip2).show();//on affiche l'image contenant les données ip2, ie la
copie de la photo avec les pixels d'intersection en blanc.
}
}
```


.10 createline

```
import java.io.*;
import ij.*;
import ij.io.*;
import ij.gui.*;
import ij.plugin.filter.PlugInFilter;
import ij.process.*;
import ij.process.ImageProcessor;
import java.awt.*;

//reporjection de points connus sur l'image, nécéssite la donnée de la calibration de l'image.
//ne prend pas encore en compte la distorsion.
public class createline_ implements PlugInFilter {

private static String defaultDirectory = null;
public ImagePlus imp;
private int compteur;
public double[] coord;
//focale(mm) mat1-1 mat1-2 mat1-3 mat2-1 mat2-2 mat2-3 mat3-1 mat3-2 mat 3-3 deltaX deltaY deltaZ
largeur(pix) hauteur orientationinterne1-1(inclus nbdepix/mm) orientationinterne1-2 centrefocalx
orientationinterne2-1 orientationinterne2-2 centrefocaly


  public int setup(String arg, ImagePlus imp) {

return DOES_32+DOES_RGB;

}



  public void run(ImageProcessor ip) {


        createline(ip);

}

  public void createline( ImageProcessor ip){
    OpenDialog od = new OpenDialog("Open coord file...", defaultDirectory);//ouverture du fichier de
calibration/orientation
    String directory = od.getDirectory();
    String name = od.getFileName();
    if (name==null)
        return;
    IJ.showStatus("Opening: " + directory + name);
    try {
      FileReader in = new FileReader(directory + name);
      readcoord( in );
      in.close();

    } catch ( Exception e ) {
      IJ.error("This does not appear to be a correct file.");

      return;
    }
    od = new OpenDialog("Open line file...", defaultDirectory);//ouverture du fichier de points à
reprojeter
    directory = od.getDirectory();
    name = od.getFileName();
    if (name==null)
        return;
    IJ.showStatus("Opening: " + directory + name);
    try {
      FileReader in = new FileReader(directory + name);
```

```
      readline( in , ip);
      in.close();

    } catch ( Exception e ) {
      IJ.error("This does not appear to be a txt file.");

      return;
    }

  }

  public void readcoord( FileReader in ) throws Exception{
      coord=new double[21];//focale(mm) mat1-1 mat1-2 mat1-3 mat2-1 mat2-2 mat2-3 mat3-1 mat3-2
mat 3-3 deltaX deltaY deltaZ largeur(pix) hauteur orientationinterne1-1(inclus nbdepix/mm)
orientationinterne1-2 centrefocalx orientationinterne2-1 orientationinterne2-2 centrefocaly
      int i=0;
    try {

    StreamTokenizer entree = new StreamTokenizer(in);

    while((entree.nextToken() == StreamTokenizer.TT_NUMBER)&&(i<coord.length))
      {
     coord[i]= (double)entree.nval;

       i++;
      }
    }
     catch ( Exception e ) {
      throw new Exception( "reading header: " + e.toString() );
    }

    return;

}


     public void readline ( FileReader in, ImageProcessor ip ) throws Exception{


      int i=0;
    try {

    StreamTokenizer entree = new StreamTokenizer(in);
    entree.nextToken();
    double X2=0;//on lit le premier point
    double Y2=0;
    double Z2=0;
    double X= (double)entree.nval;
    entree.nextToken();
    double Y= (double)entree.nval;
    entree.nextToken();
    double Z= (double)entree.nval;

    while((entree.nextToken() == StreamTokenizer.TT_NUMBER))
      {
       X2=X;
       Y2=Y;
       Z2=Z;
     X= (double)entree.nval;//on lit le nième point
       entree.nextToken();
       Y= (double)entree.nval;
       entree.nextToken();
       Z= (double)entree.nval;
       plot(X,Y,Z,X2,Y2,Z2, ip);//on trace un ligne entre le nième et le (N+1)ième point
      }
    }
     catch ( Exception e ) {
      throw new Exception( "reading header: " + e.toString() );
```



```
    }

    return;

  }
     public void plot (double X, double Y, double Z,double X2, double Y2, double Z2, ImageProcessor ip){
        //on remet les deux points en coordonnées pixels :
        double x1 = coord[1]*(X-coord[10])+coord[2]*(Y-coord[11])+coord[3]*(Z-coord[12]);//passage dans
le référentiel de l'appareil photo
        double y1 = coord[4]*(X-coord[10])+coord[5]*(Y-coord[11])+coord[6]*(Z-coord[12]);
        double z1 = coord[7]*(X-coord[10])+coord[8]*(Y-coord[11])+coord[9]*(Z-coord[12]);
        x1 = x1 /z1;
        y1 = y1 /z1;
        double xp1 =  -x1 * coord[0];//passage en coord métrique par rapport au point pricipal
        double yp1 =  -y1 * coord[0];
        double temp =  xp1*coord[15]+  yp1*coord[16]+ coord[17];//passage en coordonnées pixels
        yp1 =  xp1*coord[18]+  yp1*coord[19]+ coord[20];
        xp1= temp;
        int i = (int) xp1;
        int j = (int)(coord[14]- yp1);
        x1 = coord[1]*(X2-coord[10])+coord[2]*(Y2-coord[11])+coord[3]*(Z2-coord[12]);
        y1 = coord[4]*(X2-coord[10])+coord[5]*(Y2-coord[11])+coord[6]*(Z2-coord[12]);
        z1 = coord[7]*(X2-coord[10])+coord[8]*(Y2-coord[11])+coord[9]*(Z2-coord[12]);
        x1 = x1 /z1;
        y1 = y1 /z1;
        xp1 = - x1 * coord[0];
        yp1 =  -y1 * coord[0];
        temp =  xp1*coord[15]+  yp1*coord[16]+ coord[17];
        yp1 =  xp1*coord[18]+  yp1*coord[19]+ coord[20];
        xp1= temp;
        int i2 = (int) xp1;
        int j2 = (int)(coord[14]- yp1);
        ip.drawLine(i,j,i2,j2);//et on trace la ligne reliant les deux pixels


     }

  }
```